711-WP-001-001

Reuse Report for the ECS Project

White paper - Not intended for formal review or Government approval.

March 1996

Prepared Under Contract NAS5-60000

RESPONSIBLE ENGINEER

Audrey B. Winston /s/	3/18/96
Audrey Winston, System Engineer EOSDIS Core System Project	Date

SUBMITTED BY

M. S. Deutsch /s/	3/18/96
Michael Deutsch, Quality Office Manager EOSDIS Core System Project	Date

Hughes Information Technology Systems
Upper Marlboro, Maryland

This page intentionally left blank.

Abstract

This study of reuse for the ECS Project seeks to move the ECS project toward a component-based development paradigm for a downstream evolutionary change era. This era would be represented by Releases C and D or equivalent time-span where ECS components would be included in end-user systems. The study explores how reuse can be explored to increase the capacity to make evolutionary changes on the current system so that improved user satisfaction can be realized in this downstream era. The concept of component-based software development maximizing reuse focuses on the ability to package software into self-contained units that can be put together to build larger applications.

The end goal from a reuse perspective in preparing for the aforementioned evolutionary change era is to provide two major assets: 1) A set of reengineered core design patterns with more generalized characteristics. This will reduce later effort to implement evolutionary changes or to incorporate into later scaled system variations. 2) A component or pattern based software engineering process. A byproduct of these assets is a set of frameworks for end users who have an interest in incorporating basic ECS applications within external systems. This study has made considerable progress in defining these assets with supporting analyses and experiments.

Keywords: reuse, component-based development, evolutionary change, architecture, reengineering, design pattern, domain analysis

This page intentionally left blank.

Contents

1. Introduction

2. Reuse Study Scope and Method

3. SDPS Architecture Analysis

4. Advanced Scenarios

5. Potential Architectural Patterns

6. Case Study Experiments

7. The Pattern-Based Reuse Process

8. Conclusions and Recommendations

Appendix A

Appendix B

Abbreviations and Acronyms

Figures

Tables

1. Introduction

1.1 Purpose

This study of reuse for the ECS Project seeks to move the ECS project toward a component-based development paradigm for a downstream evolutionary change era. This era would be represented by Releases C and D or equivalent time-span where ECS components would be included in end-user systems. The study explores how reuse can be explored to increase the capacity to make evolutionary changes on the current system so that improved user satisfaction can be realized in this downstream era. While there has been no specific effort toward reevaluating present architectural concepts, it is unavoidable that reuse must take into account general information system movement toward more open, loosely coupled structuring. Some of the processes developed and used in the study have undergone experimentation in actual project groups and with actual project artifacts assessing, in some cases quantitatively, the effect of the approach on the group's activities and resources.

The concept of component-based software development maximizing reuse focuses on the ability to package software into self-contained units that can be put together to build larger applications. With component-based software development, the developer using components doesn't (or shouldn't) need to know how the components were written, but just the interfaces that they expose. Several past and current efforts, in the industry and academia to introduce effective technologies for component-based software development, have offered important lessons learned. The strength of the approach is dependent on three critical factors:

- Effectiveness of component-based techniques and tools. Effectiveness here means the infiltration of a reuse paradigm into the processes enacted in the organization.
- Implementation decisions on candidate evolutionary system changes based upon both user utility and degree of reuse of existing components and designs. The net result is a balance of emphasis on *what* the users need with *when* they receive the improvements.
- The general plan of the system should stay relatively stable or evolve smoothly if significant amounts of components are to be reused. This general plan refers to, the system's main components and their interactions which in this study will be called *the architecture of the system*.

The idea behind any successful and effective reuse effort is to match desired capabilities and existing needs to available artifacts in order to achieve the best possible coverage. The process described in Figure 1-1 provides an overview of how the development process (modeled here by using the OMT methodology steps) and the process to select reusable artifacts run parallel and interact. The arrows in the diagram show some of the major interactions.

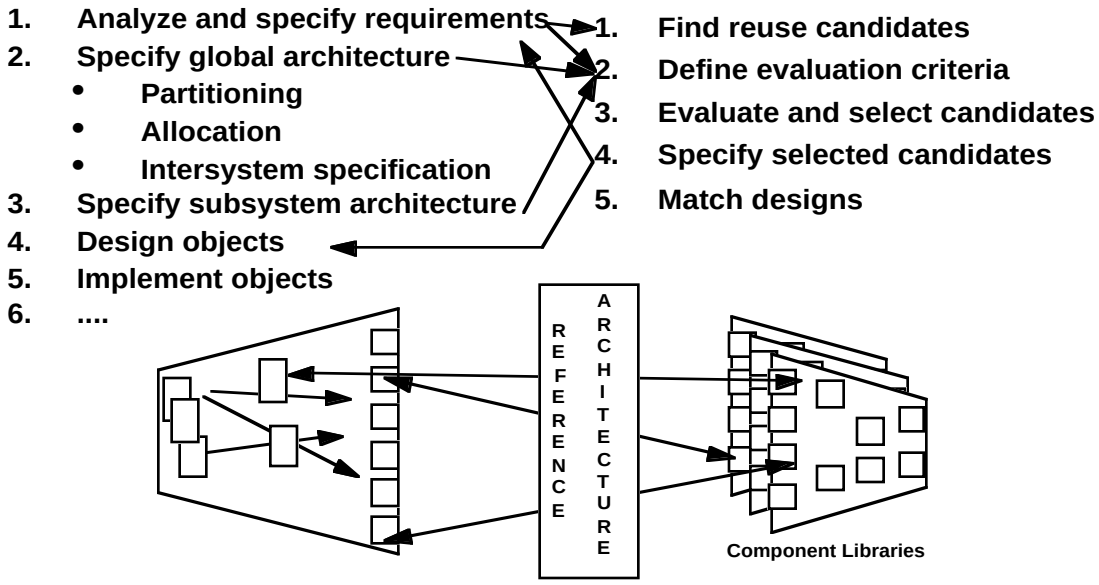


Figure 1-1. A Development Process to Reuse Components: Matching Desired Components to Available Ones

The key to the process described in Figure 1-1 is the double interaction between requirement specification and reuse candidate selection. One direction of the interaction is the obvious one in which system requirements are taken into account in the selection of reusable components; only artifacts that satisfy existing needs are considered, selected and reused. The other direction, less obvious, is the one in which the specified requirements are reviewed based on the availability of suitable artifacts; in choosing and prioritizing among requirements to be satisfied, some kind of cost benefit analysis is performed. Some requirements are satisfied and expanded because there is the availability of software artifacts that makes their satisfaction easy and cost-effective. Other requirements are satisfied in a modified way or may be deferred because there is nothing that can be reused to satisfy them; therefore they might be very expensive or, at least, not cost-effective.

The process shown in Figure 1-1 offers a motivation and an inspiring framework for the study presented in this paper. This study endeavors to responsibly project future directions where reuse can be effectively leveraged; the study makes no judgment or statement on matters of contractual scope.

Major contributors to this study have been Michael Deutsch, Show-Fune Chen, Kevin Limperos, Tom Dopplick and Dale Rickman of Hughes Information Technology Corporation, Ron Williamson of Hughes Research Laboratories, and Gianluigi Caldiera and Jyrki Kontio from the University of Maryland.

1.2 Executive Summary

The end goal from a reuse perspective in preparing for the aforementioned evolutionary change era is to provide two major assets: 1) A set of reengineered core design patterns with more generalized characteristics. This will reduce later effort to implement evolutionary changes or to incorporate into later scaled system variations. 2) A component or pattern based software

engineering process. A byproduct of these assets is a set of frameworks for end users who have an interest in incorporating basic ECS applications within external systems. This study has made considerable progress in defining these assets with supporting analyses and experiments as shown in Figure 1-2.

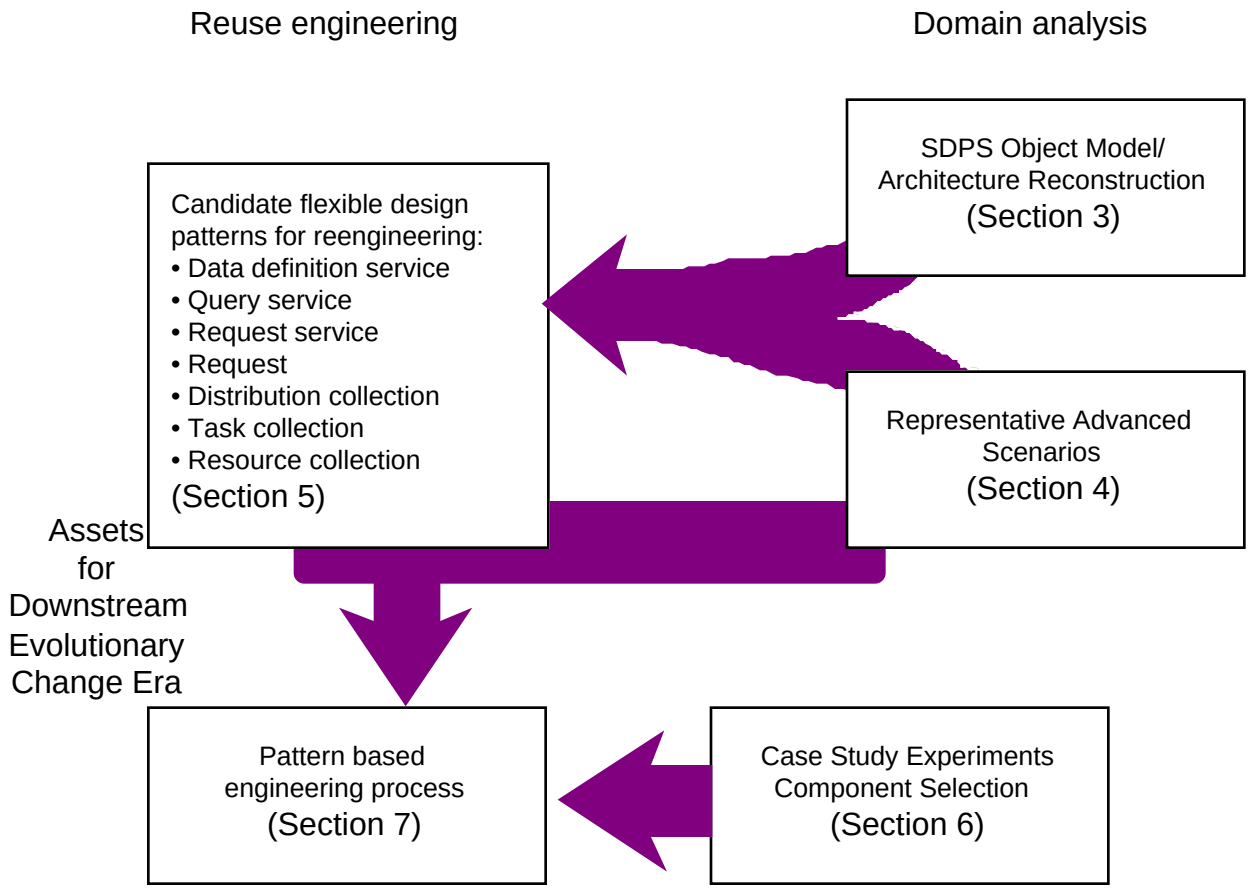


Figure 1-2. Document and Study Organization

As seen in Figure 1-2, there are two continuing major steps to this reuse study: 1) A reuse engineering activity that has defined the candidate design patterns and pattern based engineering process; and 2) A domain analysis that characterizes present artifacts, future representative scenarios, and case study experiments that shaped the definition of the reusable assets. Pointers to document sections are indicated, and a summary of each step is presented below.

The present domain, Section 3, is characterized be a reconstruction of the SDPS architecture from Release B designs. The architecture is captured in an object class model. This is the baseline from which potential more generalized design patterns can derived to support improved downstream evolutionary capacity.

Advanced scenarios, Section 4, were prepared to partially portray the domain of system usage in the downstream evolutionary era. These scenarios are intended to be representative, not exhaustive. They are partially driven by a predicted set of assumed technology factors that are reasonable extrapolations of present technology advances. Some of these scenarios possibly represent a more loosely coupled operations concept that may occur downstream. The scenarios suggest areas in the architecture where generalization could be advantageously reengineered. Scenarios are described for browse and visualization, multimedia conferencing, interactive training, on-line interactive help, on-line interactive evaluation of new tool, and interactive wireless communications.

In Section 5, reusable design abstractions are identified from the existing services for management, search, and distribution of domain specific digital products. Seven design patterns (see Figure 1-2) can be potentially reengineered to provide a more generalized capability. The design patterns are more likely to accommodate evolutionary changes with only localized impacts. Further, these patterns may be transferable into a broader context of domains similar to ECS that require these common features.

An initial component-based reuse process is offered by existing off-the-shelf software components. Little published work exists on guidelines for reusable component selection. University of Maryland personnel collaborated in experiments with Hughes Client Subsystem developers in selecting two off-the-shelf components: a map tool and a web browser tool (Section 6). The case studies indicated that a well-defined process allows the selection process to take place efficiently, the overhead of formal criteria definition is marginal, and the use of different data consolidation methods may influence the results.

In Section 7 two sub processes are outlined. The first is the process used to select reusable components in a repeatable and auditable way. The second is the process used to recognize reusable architectural design patterns starting from the design specifications. The importance of the resultant processes also lies in their applicability in different domains and circumstances.

At the end of the report some preliminary conclusion and lessons learned are presented (Section 8). Some of the lessons learned so far are:

- It is possible to consolidate some of the best practices in reusable component selection into a coherent and usable methodology which improves the efficiency and consistency of selection.
- To make the reusable component selection process repeatable, an organization needs to build up, besides software process maturity, a reuse maturity.
- It is possible to use reengineered design patterns for use in future increments of the system that involve a more loosely coupled operations concept and architectural structure.

1.3 Review and Approval

This White Paper is an informal document approved at the Office Manager level. It does not require formal Government review or approval; however, it is submitted with the intent that review and comments will be forthcoming.

The ideas expressed in this White Paper are a snapshot of the present thinking and will be considered as appropriate in both Releases C and D and in the self-governance era.

Questions regarding technical information contained within this paper should be addressed to the following ECS and/or GSFC contacts:

- ECS Contacts
  - Audrey B. Winston, System Engineer, (301) 925-0353, awinston@eos.hitc.com
  - Michael S. Deutsch, Quality Office Manager, (301) 925-0358, miked@eos.hitc.com
  - Gianluigi Caldiera, University of Maryland, (301) 405-2707, gcaldiera@cs.umd.edu
- GSFC Contacts
  - Ted Hammer, EOSDIS Product Assurance Manager, (301) 286-3295, thammer@ccmail.gsfc.nasa.gov

Questions concerning distribution or control of this document should be addressed to:

Data Management Office
The ECS Project Office
Hughes Information Technology Systems
1616 McCormick Drive
Upper Marlboro, MD 20774-5372

This page intentionally left blank.

2. Reuse Study Scope and Method

The Science Data Processing Segment (SDPS) presents multiple aspects of reuse and has been chosen as the object for this study. This segment presents reuse of internal and external components and subsystems, organized as both off-the-shelf products and specific subsystems. The other two ECS segments, Communications and System Management Segment (CSMS) and Flight Operations Segment (FOS), also present opportunities for reuse as infrastructure elements with associated challenges. This notwithstanding, it was decided to focus limited resources on SDPS as the central application area of ECS. In the following chapters it is explained how the different pieces of SDPS fit together into an organic reuse program.

2.1 Architecture Overview

The ECS architecture provides the base for an open-system design with a low cost of changing services and data types. Services and new data products can be added incrementally to the system over time, providing low cost of entry for all service providers. The service providers include Distributed Active Archive Centers (DAACs) which provide services for production of and access to standard science data products, Value-Added Providers (VAPs) which provide unique services beyond those in ECS, science computing facilities (SCFs) which provide the core Earth science data processing algorithms and conduct ongoing research, and all other related data centers.

The subsystems illustrated in the figure below capture the complete distributed functionality of the ECS spread across the nine sites illustrated in the map. The subsystems are:

- CLIENT. Provides the "client" part of the "Client / Server" access paradigm through graphical user interface and data/service access tools, as well as application program interface (API) libraries to ECS services.
- INTEROPERABILITY. Provides application level “middle-ware” which facilitates dynamic client access to services and providers holding data collections and services.
- DATA MANAGEMENT. Provides system wide distributed search and access services with multiple science discipline views of data collections and “one stop shopping” with location transparent access to those services and data.
- DATA SERVER. Provides locally optimized search, access, archive and distribution services with a science discipline view of data collections and an extensible Earth Science Data Type and Computer Science Data Type view of the archive holdings.
- DATA INGEST. Provides the “clients” for the importation of data (science products, ancillary, correlative, documents, etc.) into ECS data repositories (Data Servers) on an ad hoc or scheduled basis and deals with external system interfaces.

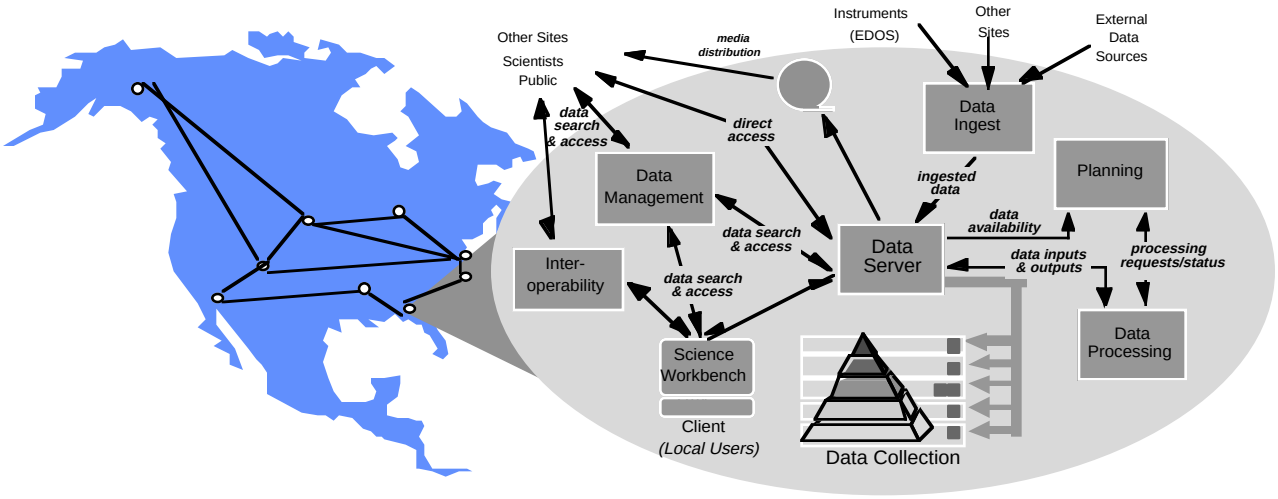


Figure 2-1. SDPS Subsystem Level Architecture

- PLANNING. Provides for pre-planning of routine / ad hoc / on-demand science data processing as well as management functions for handling deviations from the operations plans for individual DAAC sites.
- DATA PROCESSING. Provides the functions to host science algorithm software, perform data processing, process resource management and includes facilities and toolkits which offer true software portability across advanced computing platforms as well as science software integration, test and configuration management.

2.2 Presently Ongoing Reuse Activities

While this study seeks to formalize reuse for a future evolutionary change era (such as that represented by Releases C and D), reuse is being exploited on a more opportunistic basis in the Releases A and B time frame. A notable example is the reuse of the Hughes Delphi Class Libraries that furnish a generalized planning and scheduling capability.

A more formalized present reuse activity is being managed by the Development Engineering organization that is designing and implementing cross-CSCI common elements. These include scenario primitives, key mechanisms, and failure scenarios. Examples in each of these areas are noted below:

- Scenario primitives (examples)
  - L0 check
  - Create staging file
  - Submit subscription
  - Find service
  - Acquire data
  - Acquire document
  - Distributed Info Manager Result Retrieval
  - Callback

- Key mechanisms (examples)
  - Universal references
  - Error/event handling
  - Request tracking
  - Session-server-request pattern
  - Generic subscription
  - Distributed object framework
- Failure scenarios (examples)
  - Data server DBMS data corruption
  - Archive failure after user requests data
  - Server loses connection to DBMS
  - External data insert failure

The net effect of these common elements is to induce a more layered software architecture with visible interfaces that are (re)usable for both Releases A and B. The identification of these candidate common elements has appeared through ongoing experience with the architecture. It is speculated that there are other higher level larger scale abstractions that can be generalized for the benefit of more economical evolutionary change and extension. The session-server-request pattern noted above is an example of a generalized higher level design pattern that is presently clear; this study is intended to discover additional design patterns of this nature and document an associated process for their use.

2.3 Reuse Study Approach

The methodology used in the reuse study is intended to move the ECS project to the upper regions of the reuse maturity scale where the schedule and cost benefits are greater in making evolutionary extensions to the system. This progression is illustrated on Figure 2-2. Generalization of major design patterns, or mini-architectures, is envisioned as the primary vehicle.

REUSE MATURITY AND CYCLE-TIME REDUCTION

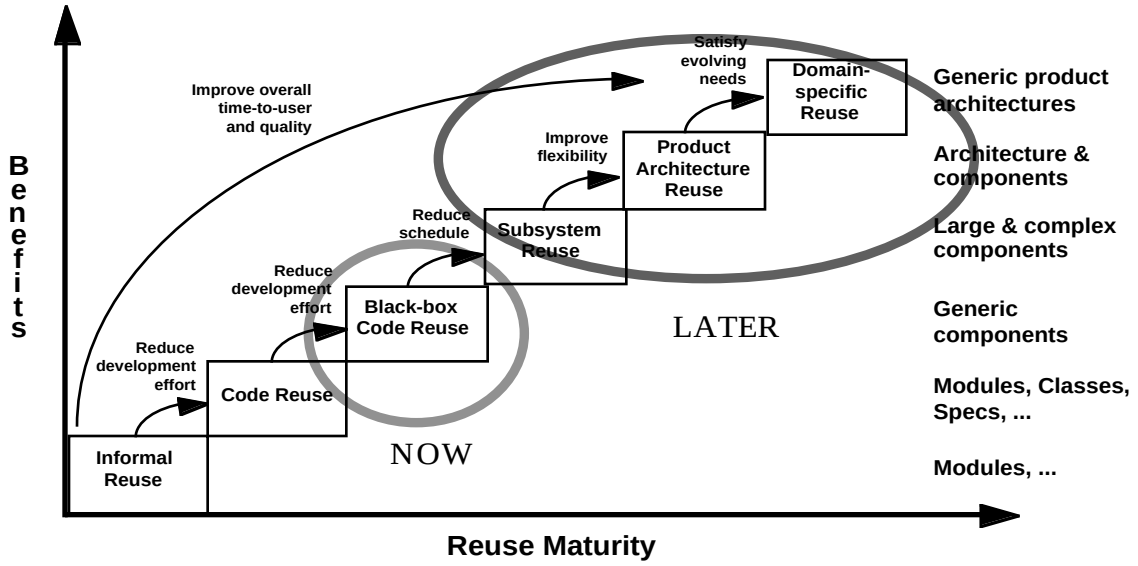


Figure 2-2. The Reuse Maturity Scale

Anecdotal evidence from the Goddard Software Engineering Laboratory and Hewlett-Packard offers affirmation of the efficacy of this strategy.

The steps of the methodology are largely an expansion of the steps previously shown in Figure 1-2 and are an extrapolation of the experience encountered in GSFC's Software Engineering Laboratory. These steps are depicted in Figures 2-3 and 2-4.

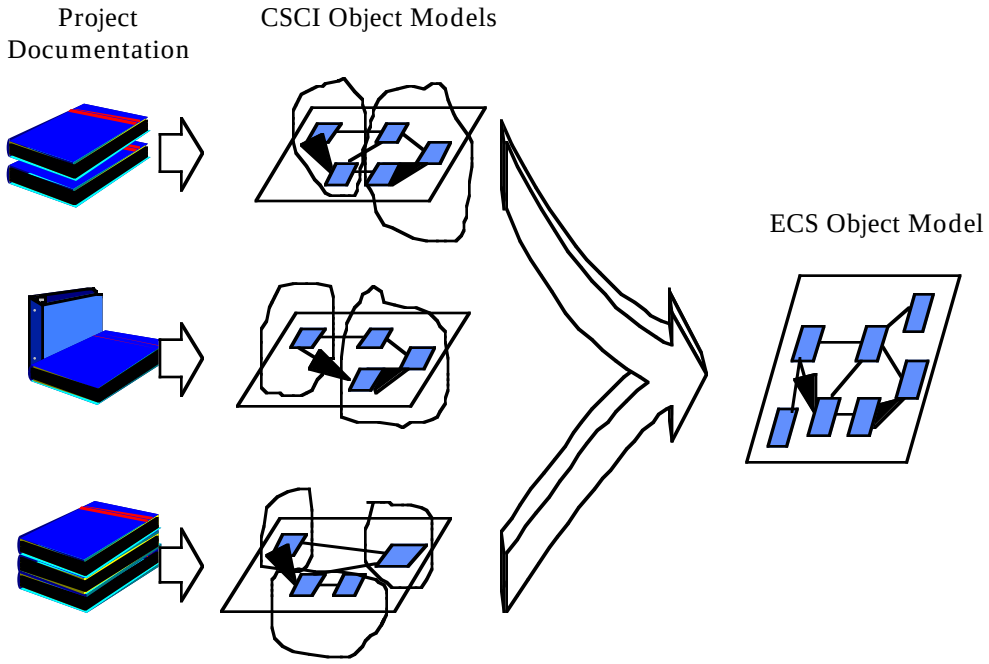


Figure 2-3. First Domain Analysis Steps

SYNOPSIS OF THE METHODOLOGY

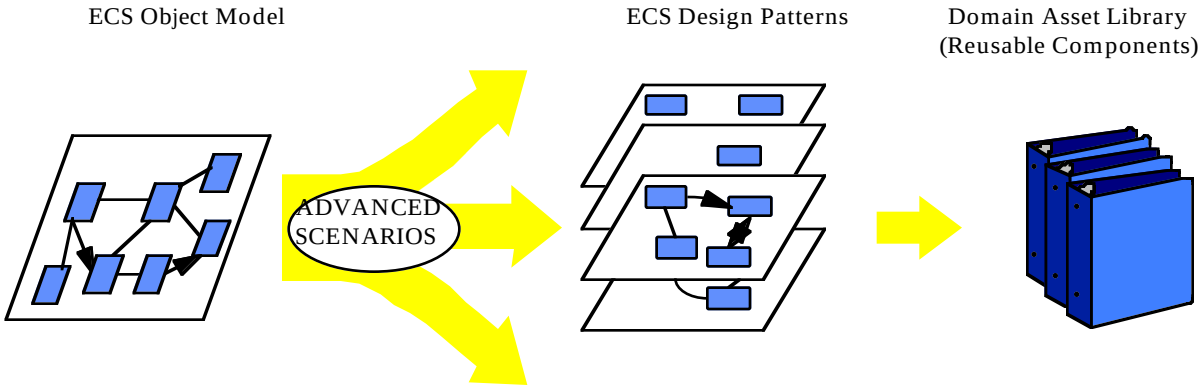


Figure 2-4. Final Domain Analysis Steps and Reuse Reengineering

The domain analysis steps entailed:

- Updating an object model for the SDPS. The building blocks used for this analysis were the CSCIs of that segment. The level of detail used was Release A Critical Design Review specifications and Release B Interim Design Review specifications.
- Use cases from speculative advanced scenarios for the ECS program were symbolically executed on the ECS object model to determine leverage areas for critical design patterns.
- Experiments in COTS selection on the Client subsystem.

The reuse reengineering steps entailed:

- Derivation of design patterns and exploration of reengineering required for generalization. An asset library describing these design patterns in terms of purpose, structure, participants and ECS source will eventually be produced.
- Documentation of a pattern based engineering process that will complement eventually the catalog of designs.

The purpose of the ECS object model and design patterns associated with it was to have a family of reusable products that are well understood and easily modifiable to satisfy the evolving needs of the ECS users. The use of both current and advanced scenarios allows us to have a proactive attitude on system evolution and satisfaction of user needs and to take full advantage of the power of the architectural point of view.

3. SDPS Architecture Analysis

3.1 The Resultant ECS Object Model

We began our analysis/diagram of the ECS model by:

- showing the relationships and associations between the subsystems,
- analyzing the use cases to determine where we get the biggest benefit from reuse, and
- better understanding the ECS system.

After all the CSCIs were examined, we compared the 12 block diagrams (refer to Appendix B, Figures B-1 to B-12) and created a generalized ECS Object Model linking all the objects together using the Rational Rose tool. Appendix A shows the resultant ECS object model using Release B documentation, as it is too large to put in the body of this document, but a silhouette of the resultant model is shown in Figure 3-1. Next we looked at scenarios that might be relevant to ECS and abstracted design patterns that should prove to be useful.

The goal of the ECS reuse study was to analyze the ECS project and determine which assets of the system can be reused downstream by focusing attention on the objects being developed now in the system. The study then followed a process which generalized the concepts so that future releases can take advantage of reusable design patterns.

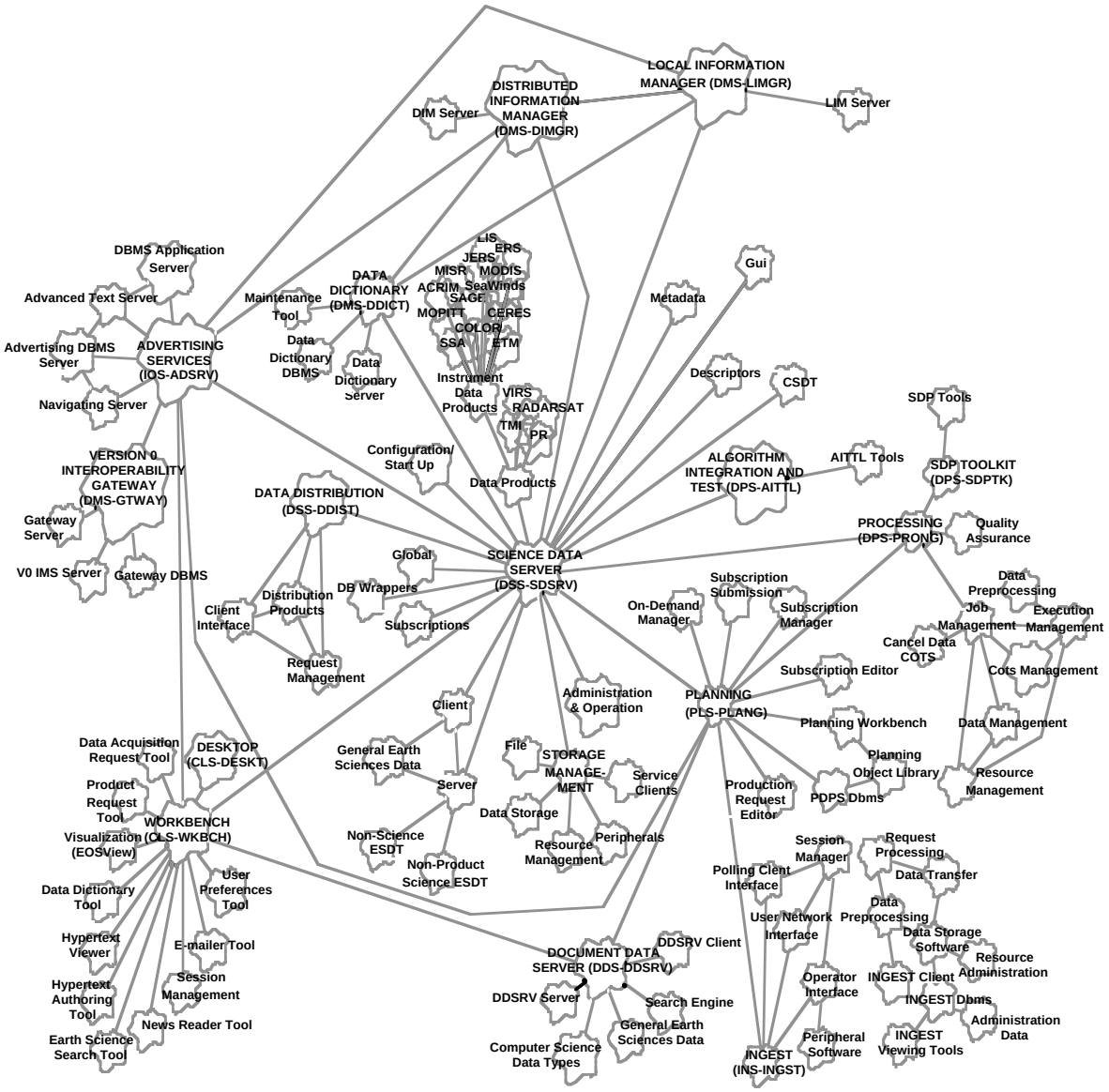


Figure 3-1. ECS Object Model

3.2 The Initial Layering and Partitioning of the ECS CSCIs

We began by looking at the TRMM (Release A) documentation, specifically the sections in Data Item Description (DID) 305 that related to the Computer Software Configuration Items (CSCIs). Here we gathered information to be used as the basis for our model. We evaluated the information down to the computer software component (CSC) level and gathered as much detail as possible resulting in twelve CSCI diagrams such as the one depicted in Figure 3-2. The full set of the twelve CSCI diagrams are located in Appendix B.

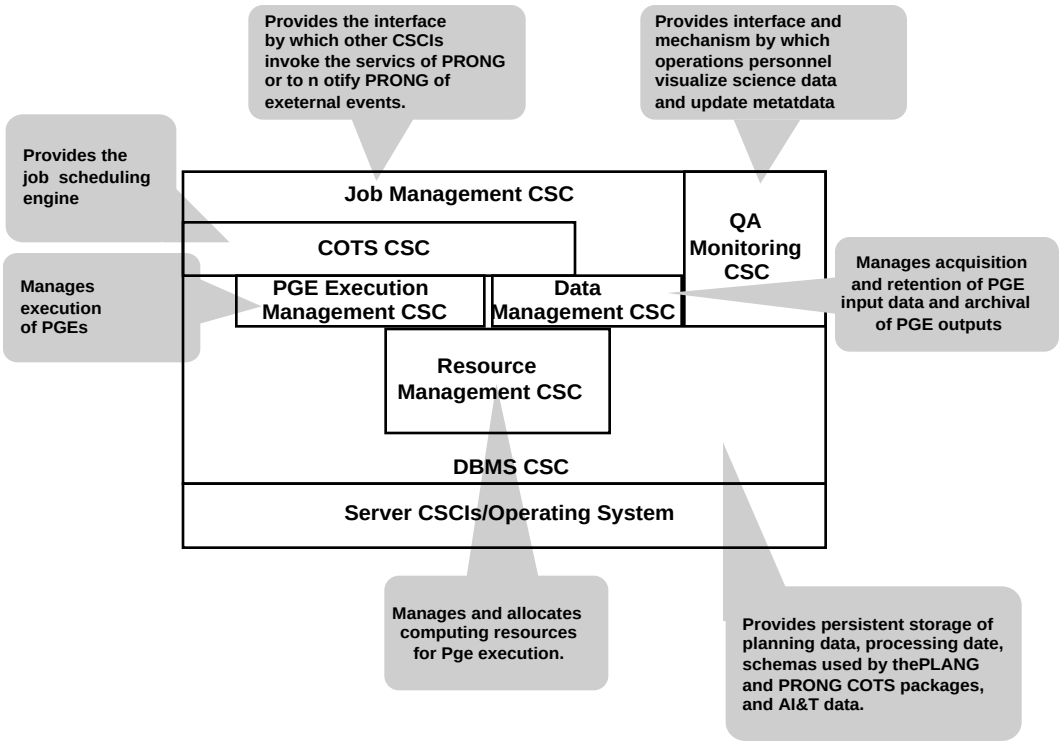


Figure 3-2. Processing CSCI

Some hints regarding layering were gleaned from the tables and documentation in DID 305. Further study of the documentation allocated specific classes to CSCs. We studied the object models in the DID and marked the classes that belong to each CSC using highlighters of different colors. The goal was to look for associations between classes from different CSCs. If there was an association, then there would be an interface between the two CSCs. If the relationship was client/server, then the CSCs would be in adjacent layers with the server below the client. If the relationship was peer-to-peer, then the CSCs would be in the same layer with a single line separating them. If two CSCs had client-server relationships with a third CSC but no relationship with one another, then they were put in the same layer but separated by a double line. For the description balloons, we pulled the text directly from tables and descriptions in DID 305. Next we deduced the layering from everything learned from the previous activities.

3.3 The Subsequent Layering and Partitioning of the ECS CSCIs

However, we soon realized that we needed to show the use relationships between the objects rather than simply the associations. So we transformed each of the earlier diagrams to the adopted Booch Object Notation to document our architectures and created CSCI architecture class category models based on the information we found; this part of the effort proved to be the crux of the exercise. An example is shown in Figure 3-3; refer to Appendix B for the full set of CSCI diagrams. The CSCs were summarized from the documentation and then we partitioned

the information in such a way as to accurately represent the CSCIs in block diagrams while at the same time keeping the reuse issue in the forefront. Throughout this exercise we noted use, inheritance and composite relationships.

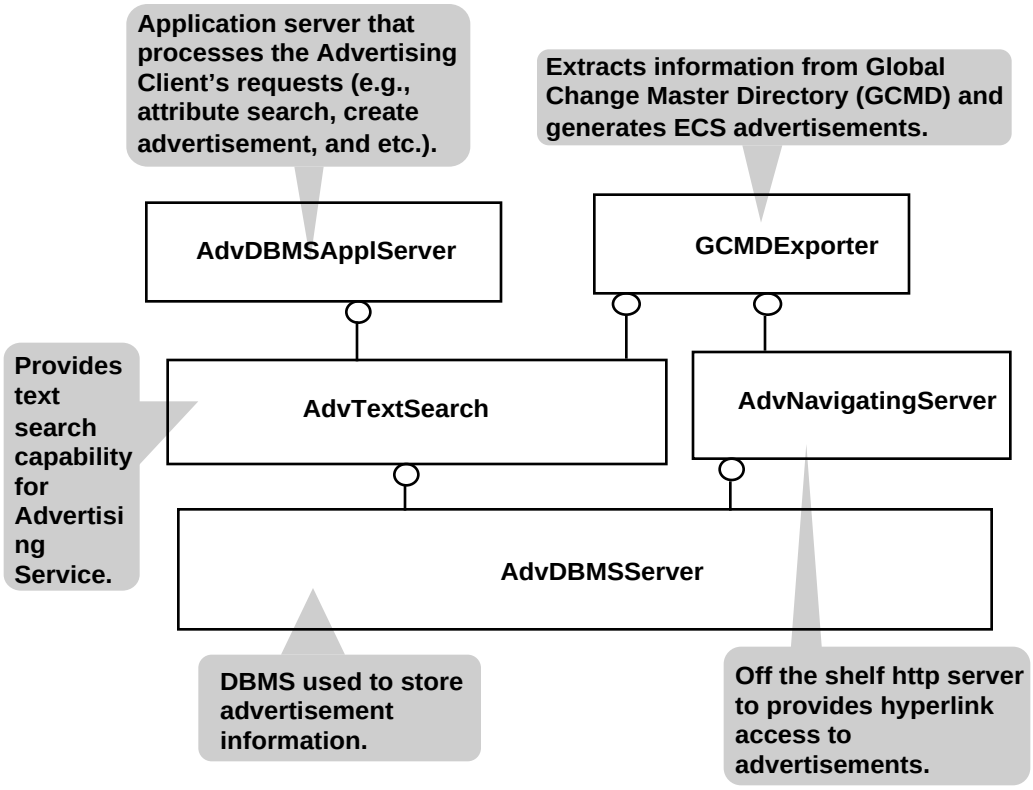


Figure 3-3. Advertising CSCI

4. Advanced Scenarios

This collection of advanced scenarios was prepared to support this study on the reuse of ECS components. These scenarios are intended to be representative, not exhaustive. They are partially driven by a predicted set of **Assumed Technology Factors** listed below that are reasonable extrapolations of present technology advances. Some of these scenarios possibly represent a more loosely coupled operations concept that may occur in a downstream era.

- Inexpensive, high powered computers
- High speed long distance communications
- Inexpensive, high-volume storage
- Mobile computing & personal digital assistants (PDA)
- Smart agents
- Continuous speech recognition
- Neural networks

Summarized in the figure below is a tabularization of these advanced scenarios against present ECS subsystems involved. These affected areas provide the first "clue" on where generalization of existing assets could furnish downstream reuse leverage. Also deduced are three potentially beneficial reengineering areas for ECS extension and generalization instigated by these representative scenarios.

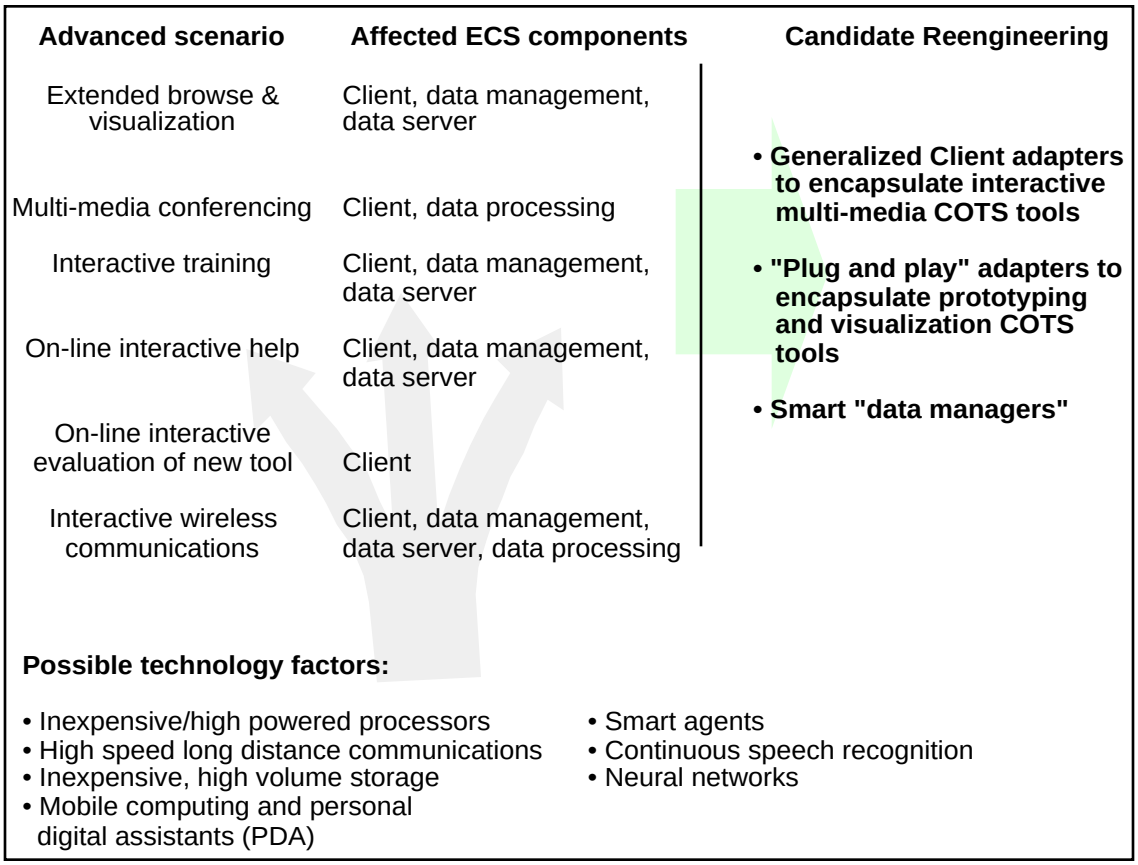


Figure 4-1. Advanced Scenarios Help Identify Candidate Reengineering to Improve Downstream Reuse

The deduced candidate reengineering areas, if implemented, are intended to answer a more widespread need to increase evolutionary change capacity with maximum reuse within reduced schedules. These specific scenarios are employed only as suggestions toward that end. More details on prospective design patterns within these candidate reengineering areas are treated in Section 5, Potential Architectural Patterns.

4.1 Scenario 1 - Browse and Visualization Scenario

A scientist studying biospheric-atmospheric interactions attempts to develop a current vegetation cover map for the continental United States. She feels that many of the EOS products developed from the surface imagers, such as MODIS, MISR and ASTER will be invaluable in her effort. The investigator decides to query ECS to see what products are available. She knows that a multitemporal MODIS NDVI product should be available, so she accesses ECS from her workstation at the university. She develops an inventory level query to locate regional products of leaf are index (LAI), net primary productive (NPP) produced annually for the previous 2 years, and level 3 NDVI products.

The response lists indicate that 100 products meet her initial criteria. To limit the number of products, she refines her query using more stringent constraints, limiting the search to include only March through November NDVI data. This more stringent query results in 75 “hits.”

She selects them all and activates the STAGE button on her desktop, which causes the archive to pre-stage the selected data sets to rapid access storage. Starting with the earliest data, she begins to scroll forward in time, noting the green up for each year. She automatically transitions from one frame to the next by simply using the mouse to scroll the data. Moving from one screen to the next requires about 2 seconds. When she skips through an adjacent screen (scrolling geographically) her longest wait is 5 seconds. When scrolling through time, the longest wait is the same, but because of the buffering scheme, she can move forward or backward in time two scenes at the 2 second update rate.

Knowing that a colleague at the EPA has recently completed a project to map the vegetation resources of the continental United States, she queries the ECS to see if the results are accessible. Finding the results in the ECS archive, she downloads a copy of the EPA vegetation data base and displays it adjacent to the NDVI images she has been examining. The high LAIs for the midwestern U.S. lead her to believe that the EPA data were developed during the height of the growing season. She scrolls back in time through earlier NDVI products to see how quickly the "green up" occurs for this area. Then scrolling forward in time through the data, she interactively delineates areas for which she will need to obtain high resolution ASTER data to determine the LAI changes for different crops.

She then repeats the process, comparing the EPA product with different parts of the NDVI products, noting the green-up of northern and central hardwood forest areas on the NDVI images. She then compares the regional LAI with the EPA data product. After this quick visual analysis of NDVI products, she concludes that they have a high degree of integrity and will be important for her research on NPP for the continental U.S. The scientist specifies electronic delivery of the image products to her account on the university system.

ECS Subsystems:

Client

Data Management

Data Server

Communications

4.2 Scenario 2 - Multimedia Conferencing Scenarios

A Principal Investigator (PI) and his Co-Investigators (Co-Is) who are studying the global hydrological cycle have run into a problem. Several Co-Is have developed algorithms for estimating snow water content using a variety of instruments and techniques. Each algorithm appears to work well on the data sets used by its developer. However, a deadline for implementing algorithms on ECS is rapidly approaching, so the best must now be selected. The PI calls for a teleconference of his Co-Is to examine and test these algorithms.

When the meeting time arrives, each scientist logs onto ECS and opens the conference window. As soon as the connection is made, a text / graphics image describing the conference appears on the screen. The PI begins the conference by listing the various candidate algorithms and associated developers. He verbally explains again the purpose of the meeting. As changes are made on his screen, the participants' workstation windows are also updated. The algorithms are examined in sequence.

The first algorithm is based on both SSM/I passive microwave data and MODIS data. The responsible Co-I begins by showing text and graphics images describing the algorithm and how it was developed. He then displays raw SSM/I and MODIS images of the area and applies his algorithm, producing an image of snow water content values. He next brings up an image of *in situ* measurements and creates a new image by taking the difference between the estimated and actual water content values. He indicates that the differences are mostly small and points out the areas of best and worst performance by circling them.

At this point, another Co-I, whose algorithm uses only SSM/I data, interrupts and asks if the algorithm being evaluated is subject to contamination by thin cirrus clouds; she suggests that the algorithm be applied to data they know were obtained in cloudy areas. She sends a SSM/I image and some matching ground truth data, while the first CO-I orders the matching MODIS data. He tries out his algorithm on the new data, and it becomes obvious that the deviations from the ground truth have increased.

This process continues, with investigators showing results of their algorithms and applying them to data sets provided by others. Finally, a consensus grows and the best algorithm is chosen for integration at the DAAC. Immediately afterwards, the participants return directly to their local work.

ECS Subsystems:

Client

Communications

4.3 Scenario 3 - Interactive Training Scenario

A second year graduate research assistant is learning how to use ECS and will use the same teleconferencing function in a distributed classroom application. An interactive training session on data searching has been scheduled with the ECS Team for this afternoon, and he wishes to participate. At the appropriate time, he logs into ECS, runs the conferencing tool, and joins the session. An ECS team instructor displays the agenda on his screen, which also appears on each student's screen. He discusses the ECS data search capability, describing each feature by example, running data searches with various metadata constraints and observing the results. The instructor's screen updates during this process also are show on each student's workstation. A series of sample problems are assigned and the students work on them. The instructor observes the problems they encounter and offers suggestions. The students log off from the system as they finish the sample problems.

ECS Subsystems:

Client

Data Management

Data Server

Communications

4.4 Scenario 4 - On-Line Interactive Help Scenario

Another graduate student has a problem using the system. His search strategy has yielded no data. After trying the help feature, he still cannot locate data he is certain exists, so he selects the SPECIAL HELP button from his menu. After a very short delay, an ECS user assistance representative at the Help Desk links to his workstation. After ECS establishes that the problem occurred during data search, the student attempts his search once more. ECS user assistance views the students search efforts on the ECS support workstation.

After seeing the screens illustrating the failed search, ECS user assistance suggests relaxing the quality constraint. The user changes the strategy to include data with a quality warning flag and then reruns the search while the expert watches on his screen. This time, the search returns a list of data sets that have not been cleared by quality assurance because of an instrument calibration question.

ECS Subsystems:

Client

Data Management

Data Server

Communications

4.5 Scenario 5 - On-Line Interactive Evaluation of New Tool Scenario

An EOS PI notices on the electronic bulletin board that the University of Illinois is making available a new visualization package. A demonstration of the package by ECS is offered to interested users. The PI logs onto the ECS and runs his conferencing tool. If desired, the PI can enable conferencing. The ECS instructor begins showing displays, and the resulting screens also appear on the workstations of all interest parties. Questions about the product are handled on-line by a University representative, and the PI sends a data set to use in the demonstration. After completing the demo, ECS personnel use the conferencing facility to obtain participant feedback. The PI is excited by the package's possibilities and orders a copy.

ECS Subsystems:

Client

Communications

4.6 Scenario 6 - Interactive Wireless Communications

Note that this scenario 6 was adapted from an earlier scenario developed on ECS.

8:20 Over coffee, the NASA EOS Program Scientist asks his computer for any new mail. Video messages are displayed on his flat screen display in her study.

8:25 While listening to his mail, one of his smart agents interrupts him mail to inform him that an Aster image over the Amazon which meets his quality criteria has just become available from ECS. He tells the computer to download the image to his NASA Workstation Server.

8:34 He asks the Server, via the Personal Digital Assistant (PDA), to update % cleared forest and Carbon release calculations.

8:40 After looking at the new calculations he requests a conference with other advisors. During the conference the calculations and images can be viewed collaboratively. Consensus is reached on reporting the new findings.

9:00 Findings, notes and selected image subsets are downloaded to his PDA. Using his PDA as a slide source, he practices his presentation and video tapes it using his computer's video camera. He then views the video on his flat screen.

11:20 While driving to lunch, his PDA informs him there are 3 new articles, since yesterday, which contain Amazon and deforestation, 20 articles with Amazon and 134 with deforestation. He requests her PDA to read aloud the 3 articles which contain both keywords.

12:00 When he arrives at the restaurant for lunch, he informs his PDA to only inform him of messages from selected colleagues. All other mail and Smart agent messages are stored. During lunch he receives 8 mail messages, his PDA stores all but only interrupts him for the one from his colleague. He responds to the message and they discuss the issue for several minutes.

1:20 He presents his updated paper "Deforestation as viewed by EOS" at the Washington, D.C. Conference on Deforestation, using his PDA and wireless communication to send the data/slides to the conference display unit. During questions additional information is needed, which gets downloaded to his PDA from his NASA server.

4:00 After the meeting, the conference chairman, requests a meeting to further discuss this topic. The meeting is set up by their PDAs negotiating a time and place. The conference room is automatically reserved.

4:30 On the drive home he dictates notes on the presentation which are stored in the PDA and distributed to his colleagues.

5:00 At home he retrieves new messages and schedules the next days activities. He sets the PDA to private mode.

ECS Subsystems:

Client

Data Management

Data Server

Data Processing

Communications

From these scenarios it is possible to reasonably focus on areas where reengineering of the ECS architecture is likely to accommodate future trends:

- smarter data management capabilities (e.g., smart agent technology),
- “plug and play” adapters to encapsulate prototyping and visualization COTS tools, and
- generalized client adapters to encapsulate multi-media COTS tools.

A further trend implied by these scenarios is more distributed generation of products. Thus, another candidate reengineering area concerns generalization of ingest processing to accommodate user driven inputs in addition to the data driven interface presently designed. These reengineering areas are further treated in the next section on architectural patterns.

This page intentionally left blank.

5. Potential Architectural Patterns

The choice of an architecture and the associated design patterns is fundamentally important in supporting the advanced scenarios. In particular the Extended Browse and Visualization scenario requires the availability of services supporting search and exploration of complex data sets archived in ECS. The state of the art in browsing and visualization tools advances much more rapidly that the basic services in the core system. This requires that the browse and visualization tools be able to negotiate interfaces, protocols, and data formats with the core system via a loosely coupled dynamic interface. Multi-media conferencing, interactive training and on-line interactive help scenarios all require the ability to dynamically link into high-bandwidth system services providing reliable communication and secure links.

Each scenario requires the negotiation of varying levels of services depending on the cost, quality, and availability of video conferencing, training, and interactive help services. The on-line interactive evaluation of a new tool scenario requires layered, open system interfacing from the tool to existing system services that provide the inputs required by the tool. Finally the interactive wireless communications scenario requires the allocation of system resources to clients and servers that will enter and leave the system context dynamically. Loose coupling of clients to servers is fundamental to this type of interaction.

5.1 Open Loosely Coupled Architectures

An open, loosely coupled architecture is a key enabling factor in the development of the ECS Release A, B, C and D system increments. ECS as a core system provides reusable components across a wider context including EOSDIS and GCDIS. This system architecture provides the context for the selection of a set of architectural design patterns that will support the effective reuse of ECS capabilities across releases and across Earth Science research disciplines.

The choice of an architecture is fundamentally important in arriving at the full benefits of reuse. This conclusion is based on applied research results of the University of Maryland and other recent research in the field of software reusability. The ECS architecture is based on an extension of the classic client - server architecture through the addition of a "middleware" layer between clients and servers. The middleware layer provides services to allow clients to find relevant services, to broker the interaction between clients and servers, to manage security, to support reliable migration of services, to negotiate protocols and data formats, to provide for scalability of system resources such as network bandwidth, processing capacity, and archival storage capacity, and to manage and administer the system as a whole.

Product Navigation was used to demonstrate the reuse potential of the ECS architecture. Product Navigation is a key set of patterns that support users in finding services associated with specific product collections archived in the ECS repositories.

5.2 Design Patterns for Reuse

Designing complex software systems is a difficult undertaking and is best accomplished by experienced designers using techniques, designs, and components that have been successfully used on previous projects. Building on prior successful applications is a cornerstone of good system engineering practices. As difficult as design complex software systems is, designing reusable software is even harder. The use of design patterns has been demonstrated (see reference Design Patterns, Elements of Reusable Object-Oriented Software) to support the design or reusable software. The goal of the effort described in this document is to find and document fundamental patterns that can be generalized to support multiple application domains. The patterns chosen demonstrate properties that will withstand evolutionary changes and incorporation into end-user systems with minimal breakage and re-engineering.

This section provides examples of design patterns within the Product Navigation context. The next section provides detail on the approach used to develop these patterns. The exemplar patterns are listed in Table 5-1 and are briefly defined. Each of these design patterns are illustrated by the object models in Figures 5-1 through 5-6. Section 5.3 now analyzes Release A and ECS assets that are candidate contributors to these design patterns.

Table 5-1. Product Navigation Candidate Design Patterns

Pattern Name	Pattern Description
Request - Transaction-Session - Server-Result	This pattern models the issuance of a query, the maintenance of state of search, and the delivery of the response back to the requester.
Query - Search Server - Data Definition - Advertisement	This pattern models the interaction of a requester object with a search engine and a supporting data dictionary.
Data Collection - Data Definition - Advertisement	This pattern models the interaction between a data collection, the definition of the data types within the collection, and the advertisement of services that are provided by the collection (such as search, browse or subset)
Data Collection - Data Type - Search Server	This pattern models the relationship between the data, applications views of the structure(schema) of that data, and a search engine that provides subsets of the data based on searchable indices associated with the view.
Request - Subscription - Event Result - Notification	This pattern models the subscription to system events associated with an action specified in the subscription and creates a result.
Request - Agent - Server	This pattern models the mediation of services between the requester and the provider of services.

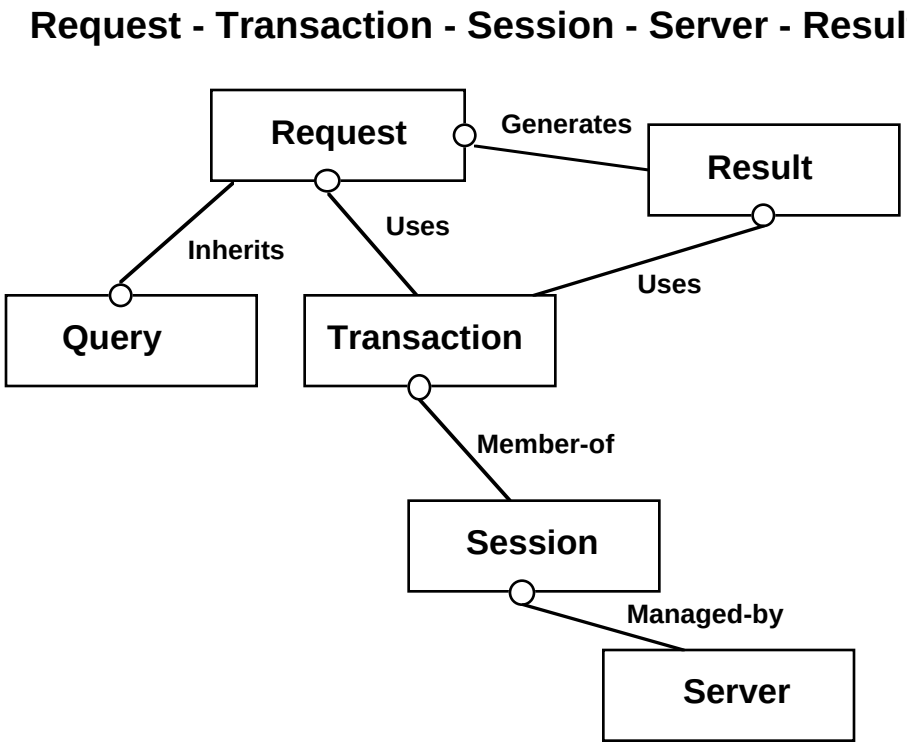


Figure 5-1. Request - Transaction-Session - Server - Result

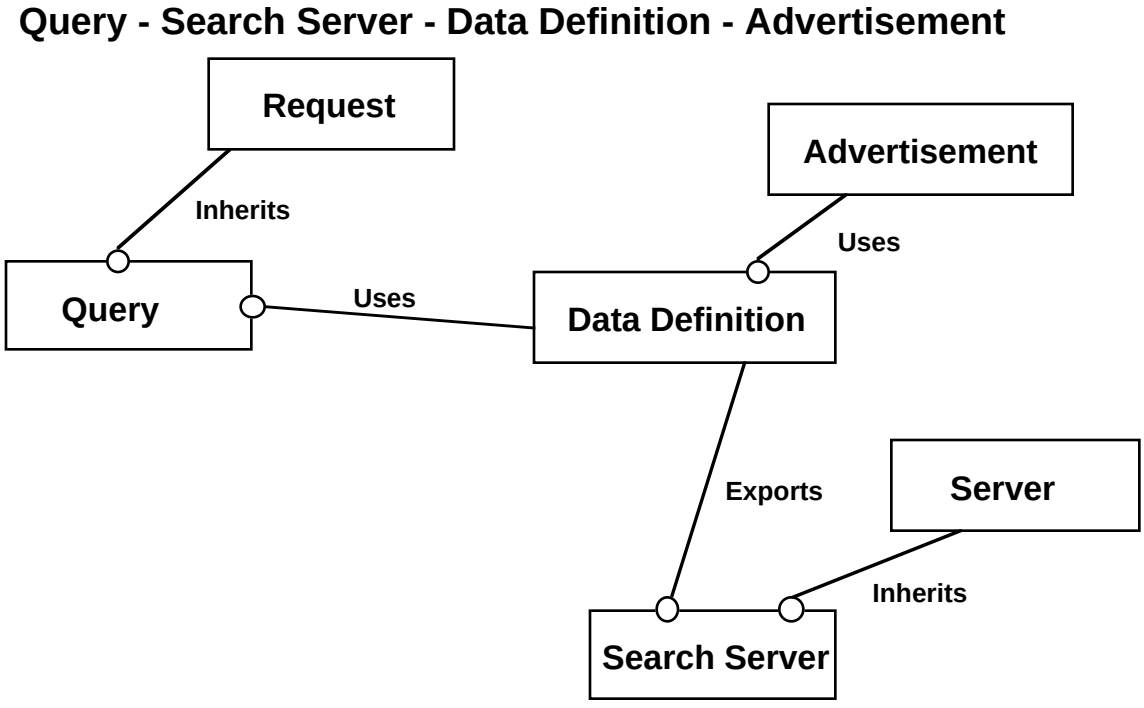


Figure 5-2. Query - Search Server - Data Definition - Advertisement

Data Collection - Data Definition - Advertisement

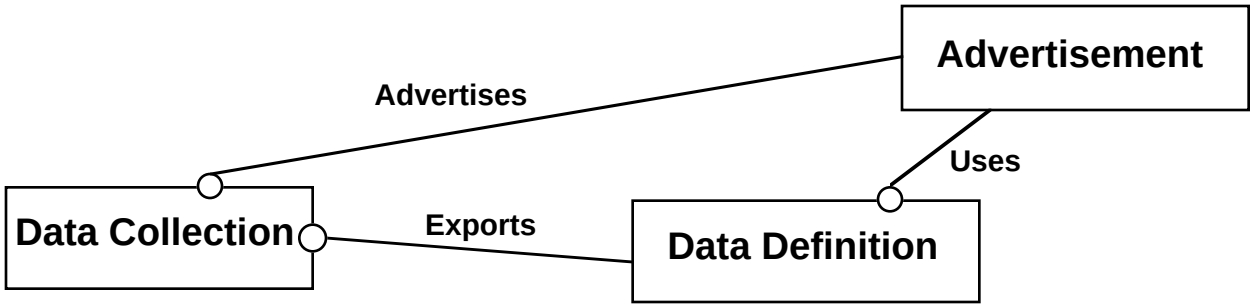


Figure 5-3. Data Collection - Data Definition - Advertisement

Data Collection - Data Type - Search Server

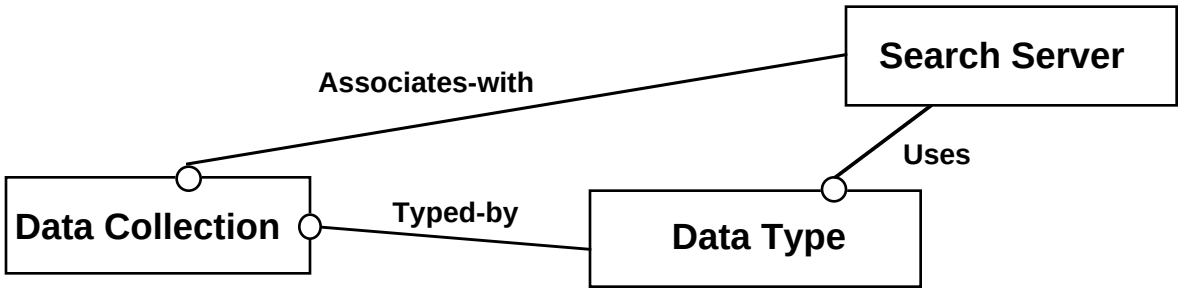


Figure 5-4. Data Collection - Data Type - Search Server

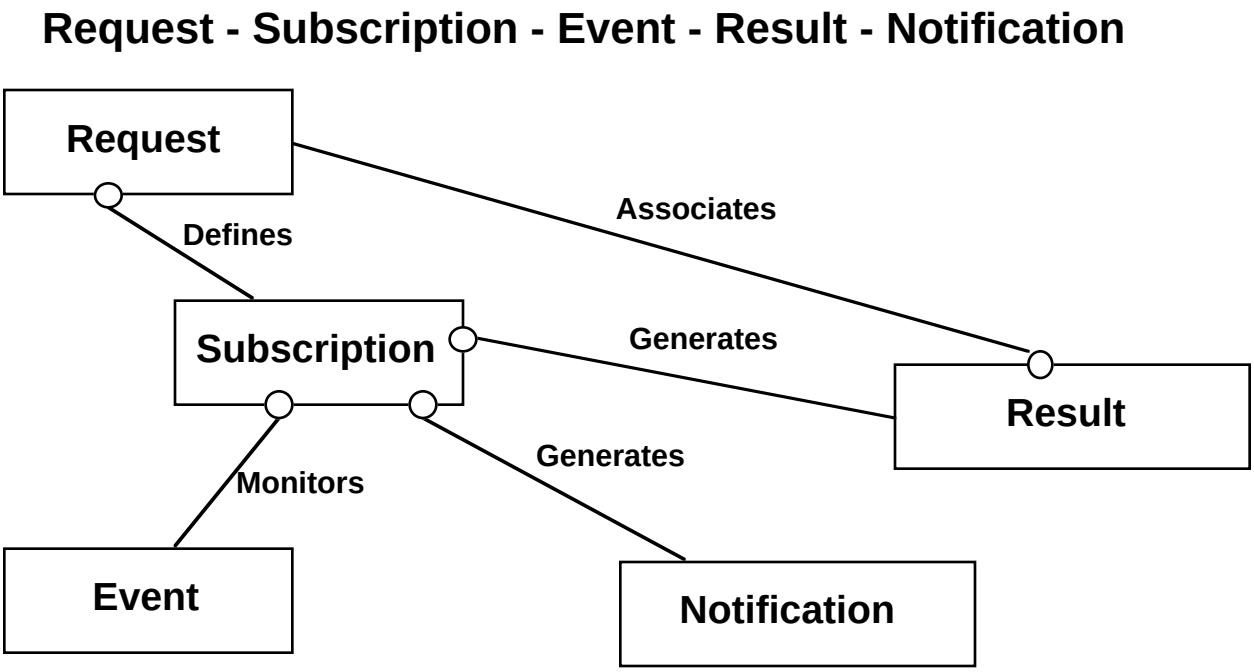


Figure 5-5. Request - Subscription - Event - Result - Notification

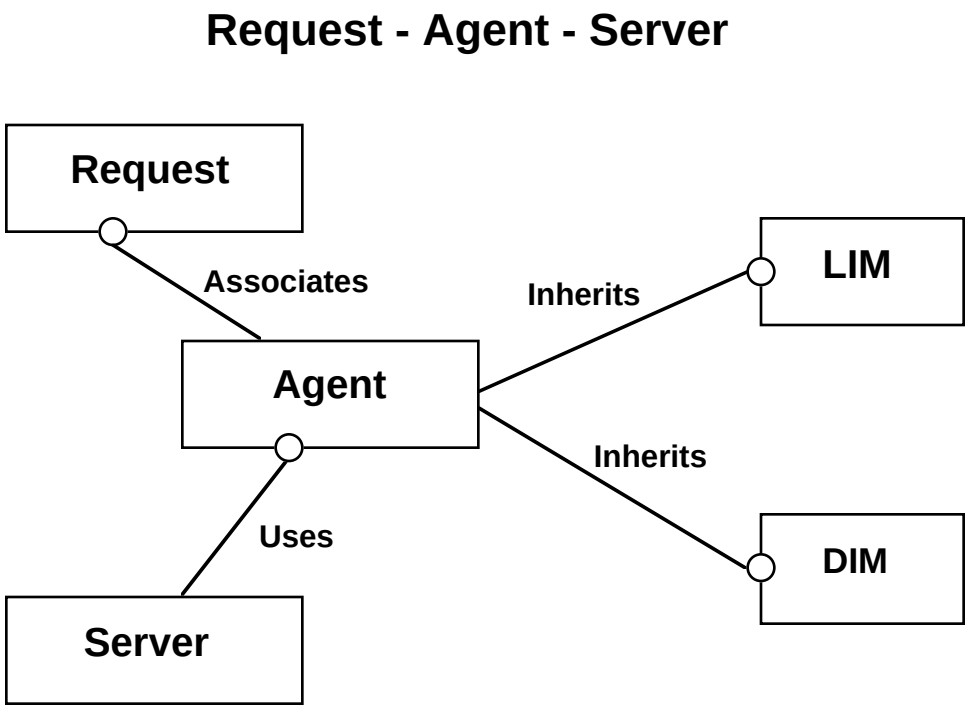


Figure 5-6. Request - Agent - Server

5.3 Product Navigation Design Patterns

The Browse and Visualization advanced scenario indicates the need for an interactive navigational access capability supporting multi- and inter-disciplinary scientists discover valuable data and relationships across multiple sensor systems. In a broader context the common browse and visualization features of each of the domains similar to ECS include the interaction by humans via a graphical user interface client; the management, access, navigation, and distribution of a large repository of digitized data; the processing of data from raw sensor data into higher level products; the reliable storage of large (terabytes) sensor and derived product data; the management of data processing resources such as communications, storage, processing, and user interfaces; and the distributed communications infrastructure underlying the entire system. This section focuses on the category labeled Product Navigation which is responsible for the management, search, and distribution of domain specific digital products. In the ECS context the Product Navigation includes those services for data management, data archiving, data distribution, and service advertising.

The ECS system design is based on a client-middleware-server architecture with objects related to the Earth Science user and application context defined in the client applications and objects responsible for storing, accessing, transforming, and distributing data collection state information stored in server applications. The client and server objects communicate via an intermediate interoperability layer (middleware) that brokers requests between the client and server objects.

The Product Navigation Design Patterns, examples of which were presented in the previous section, are a collection of object classes that cut across the Client, Interoperability, Data Management, and Science Data Server service domains in ECS. These service domains are described in the next section.

5.3.1 ECS Service Domains

In this system design context the ECS system includes several server applications each specialized to search a particular view of the ECS data holdings:

- DIM (Distributed Information Management Domain)

  The Distributed Information Manager is a search service providing a searchable federated view of the entire ECS data holdings and associated services.

- LIM (Integrated Site Information Management Domain)

  The Local Information Manager is a search service providing a searchable integrated view of a provider sites ECS data holdings and associated services.

- Data Server (Data Archive Domain)

  This provides a set of data services including the search service providing a searchable view of a logical collection of ECS data.

- Advertising Search Service (Interoperability Domain)

  This a search service providing a searchable view of ECS services and the data collections associated with those services.

- Data Dictionary Service (Data Definition Domain)

  This a search service on definitions of data elements, synonyms, constraints, valid values, and value dependencies. This service is closely linked to each of the search services above and provides a search service in its own right for the data definitions and constraints.

In each case these components take as input, from a client application, a search specification (a query in SQL, OQL, or other vendor specific extended query language) and return a result set. A session is maintained by each search service to allow the requester to obtain status and partial results from the search service. In each of the specializations of the search service described above a data dictionary is available for assistance in formulating a valid search specification. The data dictionary is coupled to the views provided by each of the search services.

Each search service is specialized both functionally (i.e. in the algorithm used to process a search request and optimize the search) and schematically (i.e. in the way it combines the data structure information from other servers, union version integration).

Figure 5-7 lists the key abstraction categories by Service Domain. These services domains are mapped onto the ECS system design and key interface and application objects are then defined and expanded as part of the ongoing preliminary and detailed designs. The following section maps the key abstractions (objects) defined in Figure 5-7 onto the services domains relevant to the Product Navigation context.

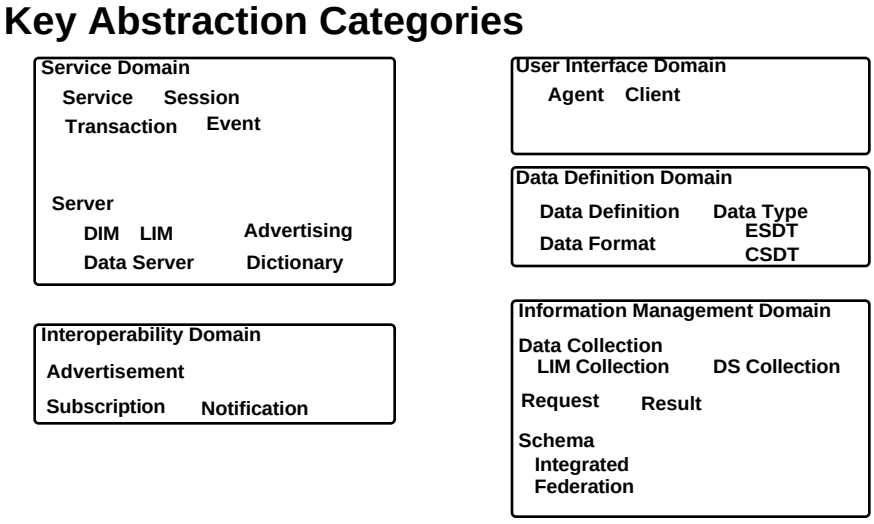


Figure 5-7. Key Abstraction Categories

5.3.2 ECS Objects within Service Domains

In the Service Domain, illustrated in Figure 5-7 above, the primary object is the Server and the associated specializations, such as the DIM, LIM, and Data Server. In the DIM service the following object classes are relevant:

- Schema Federation
- LIM Collection
- Request
- Result
- Schema
- Earth Science Data Type
- Session
- Subscription

In the LIM service the following object classes are relevant:

- Integrated Schema
- Data Server Collection
- Earth Science Data Type
- Request
- Result
- Schema
- Session
- Subscription

In the Data Server the following object classes are relevant:

- Schema
- Earth Science Data Type
- Computer Science Data Type
- Data Format
- Request
- Result
- Session
- Subscription

In the Advertising service the following object classes are relevant:

- Advertisement
- Request
- Result
- Subscription

In the Data Dictionary service the following object classes are relevant:

- Term
- Term Definition
- Term Constraint
- Request
- Result
- Subscription

5.4 Reengineering

The previous two sections provide a categorization of the ECS services and propose a generalization of those services that may be applied across a broader domain than ECS. The process of adapting the existing ECS components into a broader, more general framework is called reengineering. The reengineering task involves the adaptation of component interfaces, protocols and data formats to accommodate the broader framework.

The broader framework, for example, includes the use of middleware services to negotiate the interfaces, protocols, and data formats between a client process and the service provider's servers. Specifically, the ECS components can be generalized into the objects and services as shown in Table 5-2. These objects and services then become the core elements for the design patterns delineated in Table 5-1.

Table 5-2. Generalizations of ECS Components

ECS Component	Objects and Services
Data Dictionary Service →	Data Definition Service
(DIM, LIM, Data Server Search, Advertising Search, Data Dictionary Search) →	Query Service
Sybase and GVL Query →	Request and Query
Sybase Table and NFS File Access →	Request
Sybase Cursor →	Result
Sybase Connection →	Session
Sybase Transaction →	Transaction

The data dictionary service is a specialization of a more general data definition service. Data dictionaries in the data base world perform a very specific role in query support and optimization, the more general data definition service includes the data dictionary services and expands the role to provide a wider range of services such as valid values, dependency among valid values, linkage to service providers for the specific data type, etc.

Each of the services, DIM, LIM, Data Server, Advertising, and Data Dictionary, provide a search service which is generalized in the domain analysis into a Query Service. The Query Service in turn is then used to define a multi-pass retrieval model for access information in a variety of models, such as the Library Model or the Factory Model. The Query Service in turn is generalized into a Request Service that includes other types of services such as retrieval, browsing, subsetting, etc.

The Result, Session and Transaction objects are generalizations of database management system and network files system specific services. To minimize the vendor dependencies and provide a layer of independence for applications the database and file system specific dependency are generalized as shown in Table 5-2.

The amount of effort (E) required to reengineer the ECS services into a broader context is a function of the complexity (C) of the component (n), the number of interfaces (I), the experience of the reengineering team (X), the number of level of abstraction used to address the varied domains (L), number of layers in the application development reference model (Y) and a productivity factor (P).

$$E = P * \frac{\sum_n (C_n * I_n)}{X * Y * L}$$

From a practical perspective, each ECS Component in Table 5-2 above varies from five to twenty thousand lines of custom developed code, representing a range of 20% to 90% of the functionality of the component. For the larger components such as the Query Service, as much as 80% of the functionality is provided via COTS database management system (DBMS) facilities. The 20% custom code provides the specialization of the COTS DBMS to the ECS requirements and the encapsulation of the DBMS implementation. The 20-80 rule of thumb applies also to the reengineering effort. Certainly experimentation must be used to test the hypothesis that 20% of the existing code (in the worst case) must be reengineered to generalize the functionality to a broader domain.

A complicating factor in any information intensive system development effort is the amount of analysis, design, and development effort that must go into the information modeling and the schema specification. The information model represents the static structure of the persistent data in a system. The functionality represented by application code is coupled to varying degrees with the information model and the resulting exported schema specifications.

Traditionally, the information model has been separated from the logic of the application programs to provide a degree of independence of the logic from the data structure. The independence helps to maximize the productivity during the development phase of a project. Experience has shown that this separation leads to configuration management problems and high costs during the maintenance and evolutionary stages of a system.

The object oriented design approach favors integrating the data model and logic of an application into a unified object model. This combined approach reduces overall maintenance costs and simplifies the design pattern reengineering task but complicates the joint development.

The ECS development effort is a hybrid development effort. Both an object model and an information model have been developed and are being reconciled for each major release. Estimating the amount of reengineering effort necessary will require an analysis of both the object model and the information model using the factors define above.

5.5 Suggested Reading

ECS, 1994: Summary of the ECS System Design Specification. http://edhs1.gsfc.nasa.gov

Elkington, M., Meyer, R. and McConaughy, G., 1994: Defining the Architecture of EOSDIS to Facilitate Extension to a Wider Data Information System. ISPRS'94, Ottawa.

Gamma, E, Helm, R, Johnson, R, Vlissides, J., 1995, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley.

Moxon, B. 1993: 19300611 Science-based System Architecture Drivers for the ECS Project. URL: http://edhs1.gsfc.nasa.gov.

Williamson, R. 1995: EOSDIS Core System: A Strategy for Ground System Cost Reduction, RAL Conference Paper, September, 1995.

This page intentionally left blank.

6. Case Study Experiments

We conducted two case study experiments in this program. Their purpose was to formulate and evaluate a systematic method for reusable component selection. The case studies were performed within normal project work in the program. The case studies resulted in the selection method Off the Shelf Option (OTSO), which will be explained in detail later in this document, and assisted the program in making the reusable component selections. University of Maryland personnel worked in-line with ECS Client subsystem developers during this activity.

6.1 ReMap

The objective of our first case study, the ReMap project, was to select an OTS package for the "map application" in the ECS system. The map application is an application that allows users define areas graphically on earth's surface. These areas are used to select data in the EOS database.

The main alternatives for the map application were the rudimentary demonstration prototype developed earlier in the project, a product, UIT, from European Space Agency, and two commercial products, Delphi and STK. However, it became apparent early in the evaluation process that UIT was superior to other alternatives in terms of matching the required functionality. Therefore, an official decision to select UIT was done before the OTSO method was thoroughly carried out. As we wanted to test our method in any case, we conducted a "simulation" of the process to the end, i.e., we conducted a limited analysis of all alternatives using most of the criteria and carried out financial and qualitative comparisons. While the reliability of the results in such a limited case study is low, this allowed us to try out the OTSO process. Nevertheless, the outcome of our analysis seemed to support the early decision made in the project.

The ReMap project used two primary sets of criteria in the selection: functional requirements and product quality characteristics. Figure 6-1 presents the product quality characteristics that were used in the project. Note that the Figure 6-1 does not contain all the evaluation criteria used for the ReMap project. Full definitions of the criteria and evaluation results are documented separately [Kontio, J. and Chen, S. Hypertext Document Viewing Tool Trade Study: Summary of Evaluation Results, 1995. ECS project Technical Paper 441-TP-002-001, Hughes Information Technology Corporation.].

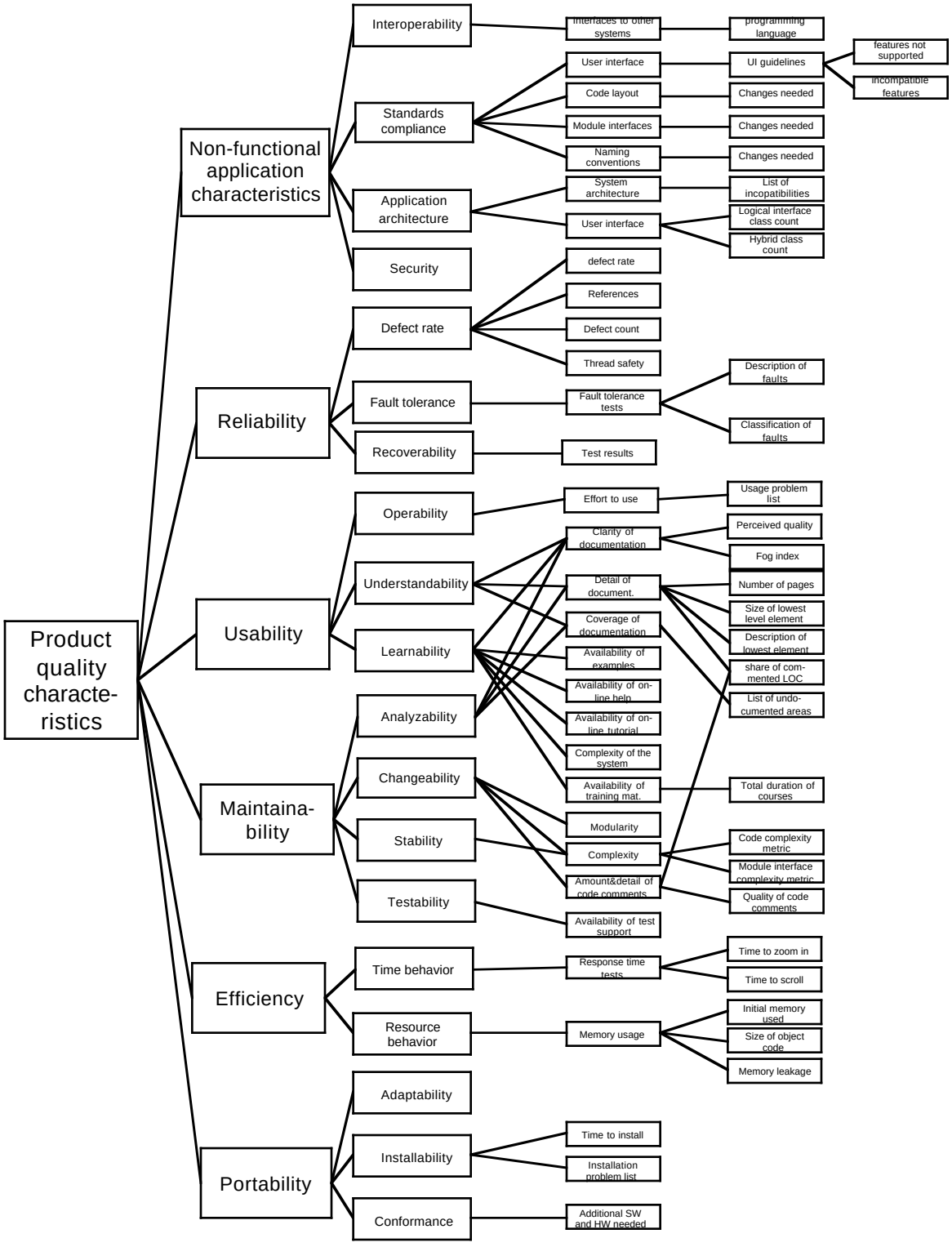


Figure 6-1. Product Quality Characteristics in the ReMap Project

In addition to providing feedback to the development of the OTSO method, the ReMap project lead us to make some other observations that are useful in similar selection processes. First, it was necessary to refine the stated requirements significantly in order to develop a meaningful evaluation criteria set. We believe that this is a common phenomenon. When the reusable software selection takes place, requirements are typically not defined in much detail. Yet detailed requirement definitions are necessary for evaluating different products. The interaction of the reusable software selection process and requirements definition process is essential. We also witnessed that evaluating reuse alternatives not only helped in *refining* the requirements, it also led to *extending* the requirements in some situations. This is an additional challenge in requirements management. In our case the extended requirements were limited to the area of the COTS but it is also quite conceivable that the evaluation process influences the whole system requirements.

Second, a considerable amount of calendar time may need to be spent on installation and logistics before the evaluation can commence. In our case, this limited the time available for detailed evaluation. If the reusable OTS software selection is in the critical path in the project, some attention needs to be placed on the logistics and procurement so that unnecessary delays are avoided. Overall, it seems that the actual effort spent on evaluating each alternative was not very high but the calendar time elapsed was.

The third observation in the project was that project personnel were unsure about where to look for alternatives and when to stop the search. As a result, the OTSO method contains some guidelines for concluding a search.

Finally, despite all the efforts in criteria definition and evaluation, there will inevitably be some data that is missing. This may be because the data is simply not available, because it would be too costly to obtain the data or the data is not available in time.

6.2 Hypertext Browser Selection

The objectives of the second case study [J. Kontio, S. Chen, K. Limperos, R. Tesoriero, G. Caldiera, and M. S. Deutsch: A COTS Selection Method and Experiences of Its Use, 1995. Proceedings of the 20th Annual Software Engineering Workshop. NASA. Greenbelt, Maryland.] were to: 1) validate the feasibility of the evaluation criteria definition approach in the OTSO method and 2) compare two different methods for analyzing the evaluation data. Our hypotheses were that the more detailed evaluation criteria definition will result in more effective, consistent and reliable evaluation process. We also expected that the multiple criteria decision support method would give decision makers more confidence in the decisions they made than the conventional weighted scoring method (WSM).

A total of over 48 tools were found during the search for possible tools. Based on the screening criteria, four of them were selected for hands on evaluation: Mosaic for X, Netscape, Webworks for Mosaic, and HotJava. These tools were each evaluated by two independent evaluators, most of whom evaluated two tools. This arrangement allowed each evaluator to have more than one reference point and enabled discussion and comparison of tools. All evaluators were Hughes

project personnel and two of them acted as “key evaluators”, i.e., they had previous experience in off-the-shelf software selection and they coordinated much of the evaluation and analysis in the case study.

The evaluation criteria were derived from existing, broad requirements. However, it was soon discovered that the level of detail in the documented requirements was clearly insufficient for detailed technical evaluation of the software selected. The requirements had to be elaborated and detailed substantially during this process.

The hands-on evaluation was based on the evaluation criteria defined earlier and evaluators wrote reports addressing all the evaluation criteria for each tool. The results were discussed in a joint meeting with all evaluators, and all open issues or conflicting evaluation results were logged and assigned as action items.

In the joint evaluation meeting the differences between tools were first discussed qualitatively. The scores for the WSM were also assigned for each tool were assigned after this discussion. In most cases only the “high” and “low” values were explicitly defined for the scores verbally before assigning a score. Note that the scores were, therefore, mostly based on ordinal scales.

The qualitative differences between tools were documented separately [Kontio, J. and Chen, S. Hypertext Document Viewing Tool Trade Study: Summary of Evaluation Results, 1995. ECS project Technical Paper 441-TP-002-001, Hughes Information Technology Corporation.]. This report represents the “raw” differences between the tools without any judgments of their importance or ranking to each other.

After the evaluation meeting, we waited for two weeks before continuing with the analysis, partially because of the logistics of completing the action items on missing information and partially to allow the two key evaluators to “forget” the numerical scores that were assigned to alternatives.

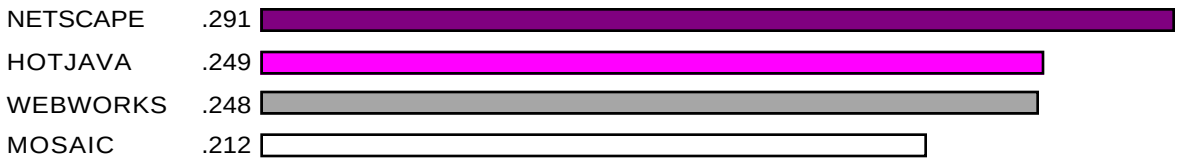


Figure 6-2. Results of the Analysis (AHP Method)

For multiple criteria decision support we used the Analytic Hierarchy Process (AHP) method and a supporting tool [T. L. Saaty, Expert Choice software 1995, ver. 9, rel. 1995. Expert Choice Inc. IBM. DOS.]. The resulting scores for the four tools using the AHP analysis method are presented in Figure 6-2. The bars in Figure 6-2 represent the relative preferences for these tools.

The total effort distribution for the evaluation process is presented in Table 6-1. After the case study was completed, we interviewed the key evaluators for their experiences, observations and perception of the process. Specifically, they were asked to state how useful they found the criteria definition process and the two analysis methods used.

As far as the OTSO criteria definition process is concerned, we noticed that the effort spent in criteria definition, 40 hours (i.e., 28% of the total effort) was within the range we expected. According to our interviews with the evaluators, it clearly had a positive impact in the efficiency, consistency and quality of evaluations. Evaluators were given a well-defined template that allowed them to focus on important characteristics of the tools, they were able to discuss the features with each other and evaluation results were relatively consistent. However, despite our efforts, there were still some vague definitions for some of the criteria: a couple of criteria definitions were misunderstood and one was found to be irrelevant during evaluation.

Table 6-1. Effort Distribution in the OTSO Case Study

Activity	Effort (hours)		%	
Search		**20**		**14%**
Screening		**8**		**5%**
Evaluation		**79**		**55%**
Criteria Definition	40		28%	
Mosaic for X	10		7%	
Netscape	9		6%	
Webworks	9.5		7%	
HotJava	10.5		7%	
Analysis/WSM		**5**		**3%**
Analysis/AHP		**7**		**5%**
Management/Administration (planning meetings, reporting, etc.)		**20**		**14%**
Learning about the Methods and Techniques		**1**		**1%**
Other (vendor contacts, installations)		**4**		**3%**
Total		**144**		

The comparison of WSM and AHP methods supported our hypothesis. The results of the WSM seemed reasonable for the evaluators but the WSM table did not provide any insights to the sensitivity of the results. Also, with 38 criteria, it was rather difficult to see the big picture in the data.

The AHP method was initially perceived as difficult as its calculation model is more complex and it involves a high number of paired comparisons. In fact, during the evaluation we made 322 such comparisons, not including revisions made to some comparisons. This would not have been practical without a graphical, easy to use tool that allowed this to take place within a single session.

The large volume of individual assessments in the AHP method is perhaps its main weakness. Even though the overall duration of the assessment session was not too long, the repetitive assessments may cause fatigue in evaluators. Fortunately, the paired assessment method automatically produces information on how consistent the evaluations are, which would be the

likely result of fatigue. The moderator can monitor the consistency, as we did in our case study, and call a break if consistency rates become alarming.

The AHP method allowed the key evaluators to analyze the results from various perspectives and play what-if scenarios by changing weights for different criteria groups. Also, the redundancy built into the paired ranking method and possibility to get feedback on the consistency of comparisons further increased the confidence in results. As the AHP method produced ratio scale rankings, the method produced more information for decision making. The key evaluators agreed that the AHP method produced more information and more reliable information for decision making.

The surprising result in our case study was that the relative ranking of browsers was different using the two analysis methods. Both methods ranked Netscape as the best alternative, but the remaining order of tools was different. WSM ranked HotJava the worst but AHP ranked it the second, although very close to Webworks. The WSM could not differentiate between Webworks and Mosaic for X but AHP found clear differences between them.

Given the serious limitations of the WSM approach, we believe that the AHP results are more reliable and they better represent the real rankings between the tools. We draw this conclusion primarily based on the confidence the evaluators had with the analysis results. Unfortunately, it is practically impossible to verify this conclusions with certainty. As in all multiple criteria decision making situations involving future behavior of a system, preferences change over time and between initial information may have contained errors, individuals, situations and information available may change and objectives may also change.

The case studies support our initial hypothesis that the formalization of the reusable component selection process is beneficial. The case studies also highlight the interaction between the selection process and requirements definition, a phenomenon outlined in the engineering paradigm illustrated in Figure 1-1. It also gives insights on the relationship between system architecture and reuse: the architectural implications of reusable components should be explicitly considered in the reusable component selection process as well as in the architectural design process.

7. The Pattern-Based Reuse Process

7.1 Overview of this Section

The purpose of this section is to provide an overview of the approach developed and used in the ECS reuse study.

A reuse program is aimed at selecting software artifacts originating from one project and using them with minor modifications in another project. This also includes repackaging for downstream releases of the same project. These artifacts are in some cases already packaged as object libraries or as mini-architectures called design patterns. In other cases the artifacts are not available in reusable form and some reverse engineering is needed on an existing system in order to identify them.

Both cases have occurred and been dealt with in the ECS reuse study. On one hand, the study has focused on the issue of selecting packaged components. A process was specified and criteria were formulated to perform the selection of a COTS object library. Section 6 has already presented in some detail the experiments. This section will focus on the process derived from those experiments and specified for further application. On the other hand, the ECS reuse study has focused on the extraction of potentially reusable design patterns from the ECS system itself. Section 4 and 5 have presented examples of the scenarios used and some derived patterns. This section will present the conceptual framework and the concepts used to derive those patterns.

We have grouped those processes under the general name of pattern-based reuse process because in both cases the purpose of the analysis is to match what is needed with what is available, i.e., use recurrent solutions (patterns) to satisfy recurrent needs. As shown in Figure 1-1, a reuse program supports the software development process by identifying the needs which have a more general nature and satisfying them with reusable artifacts. The two processes presented in this section address this paradigm and provide a repeatable way of doing so.

The repeatability of the processes is ensured by:

- the explicit statement of the selection criteria,
- the general nature of the abstraction process used to identify architectural design patterns, and
- the independence from specific design methodologies, as long as they are based on the object oriented paradigm.

Figure 7-1 shows the role of the two processes presented in this section in the more general context of a reuse program.

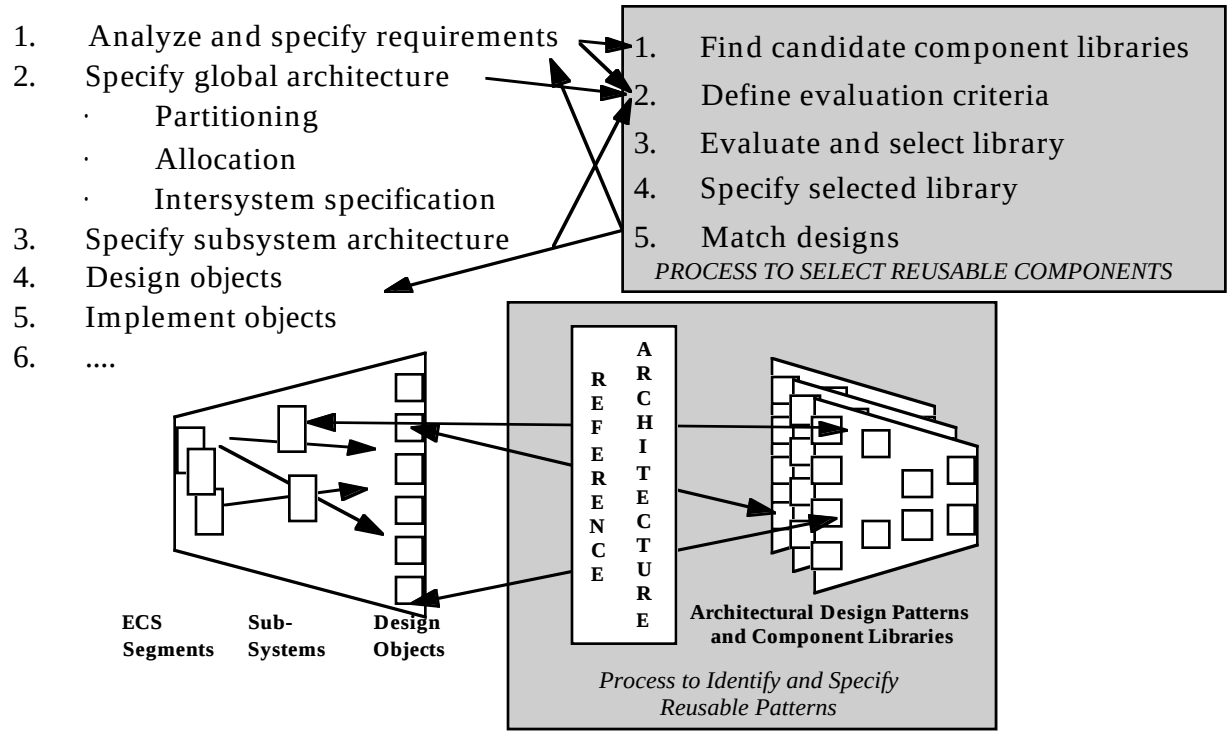


Figure 7-1. A General Development Process to Reuse Component

The materials presented in this section are extremely relevant for the implementation of these reuse strategies in terms of both methodology to obtain design patterns and process used to support and maintain them.

7.2 The Process to Select Reusable Components

The OTSO method was developed to facilitate a systematic, repeatable and requirements-driven COTS software selection process [Kontio, J.: OTSO: A Systematic Process for Reusable Software Component Selection. University of Maryland Technical Reports. College Park, MD: University of Maryland. CS-TR-3478, UMIACS-TR-95-63, 1995.]. The main principles of the OTSO method are the following:

- a well-defined, systematic process that covers the whole reusable component selection process,
- a systematic method for deriving detailed COTS software evaluation criteria from reuse goals,
- a method for estimating the relative effort or cost-benefits of different alternatives, and
- a method for comparing the “non-financial” aspects of alternatives, including situations involving multiple criteria.

The overall phases of COTS software selection are presented in Figure 7-2. The horizontal axis in Figure 7-2 represents the progress of the evaluation (i.e., time) and vertical axis the number of alternatives considered at each phase. Starting by the *search* phase, the number of possible alternatives may grow quite rapidly. The most potential candidates will need to be sorted out (*screening*) to pick the ones that can be evaluated in more detail with the resources available. Detailed *evaluation* of a limited number of alternatives determines how well each of the alternatives meets the evaluation criteria. These results are systematically documented. We have separated out the *analysis* phase to emphasize the importance of interpreting evaluation data. Sometimes it may be possible to make straight-forward conclusions if one of the alternatives is clearly superior to others. However, in most cases it is necessary to use systematic multiple criteria decision making techniques to arrive at a decision. Based on the decisions made, typically one of the alternatives is selected and *deployed*. Finally, in order to improve the selection process and to provide feedback on potential further reuse of the component, it is necessary to *assess* the success of the reuse component used in a project.

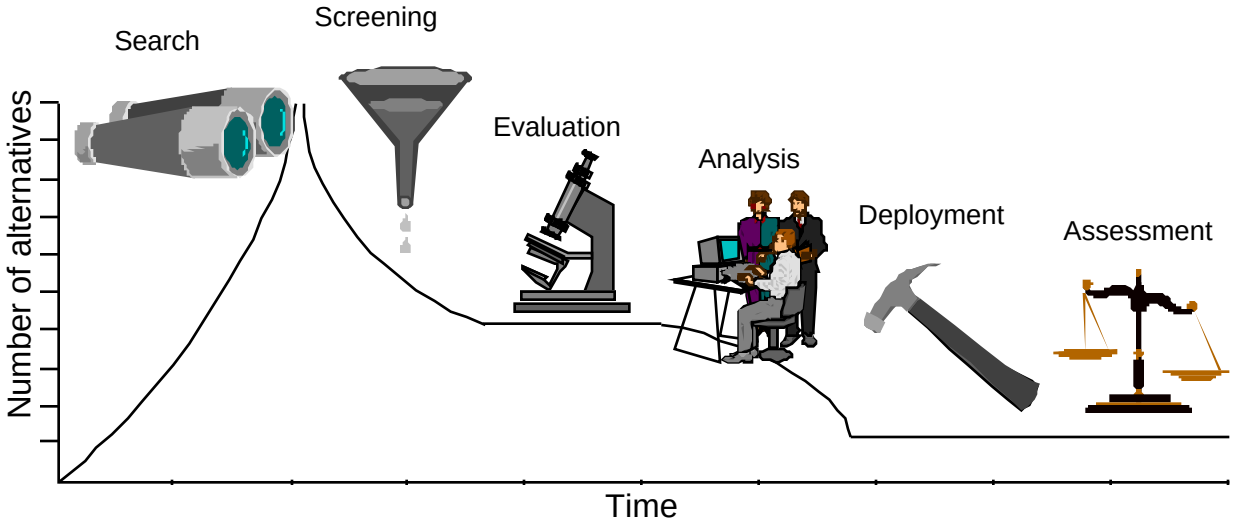


Figure 7-2. The Phases in COTS Selection

Figure 7-2 presents a high level, sequential view of the OTSO selection process. In Figure 7-3 we have presented a more realistic and detailed view of the OTSO process, using a data flow diagram notation. Figure 7-3 highlights the central role of evaluation criteria definition. In our method, the evaluation criteria are gradually defined as selection process progresses. The evaluation criteria are derived from reuse goals and factors that influence these goals.

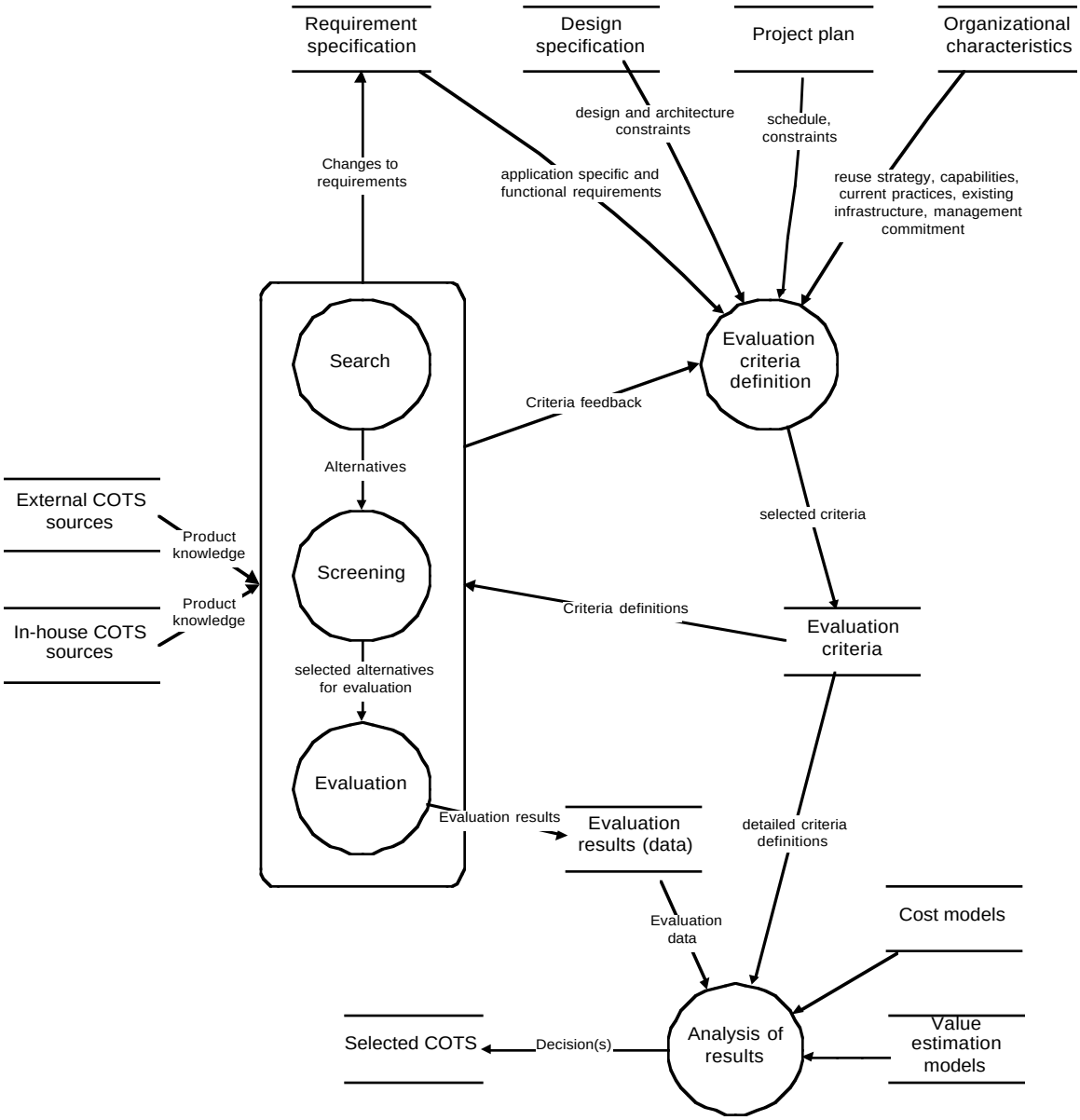


Figure 7-3. The OTSO Selection Process

The OTSO method was developed to consolidate some of the best practices we have been able to identify for COTS selection. We believe that the primary benefit of formalizing the COTS selection process is that it allows further improvement of it. A repeatable process supports learning through experience.

Our case studies were intended to provide practical experience in applying the method and to provide some indication of its feasibility in practice. The case studies seemed to suggest that the method is practical and it may improve the COTS selection process if it is currently conducted in an ad hoc manner.

In particular, the requirements driven, detailed evaluation criteria definition seemed to have a positive impact on the evaluation process. Furthermore, the marginal cost of more formal criteria definition seems to be within acceptable range as it can be accomplished within person days of effort. It also can have a positive effect on the definition of the application requirements.

Second, the case studies showed that the multiple criteria decision support method can produce more relevant information for COTS selection and this information is perceived as more reliable by decision makers. At the same time, the additional cost of applying multiple criteria decision support is small, compared to the WSM approach. However, when the number of alternatives and criteria are small, WSP may still be a reasonable method to use, provided that its limitations are taken into account and compensated.

Third, our case study also showed that the choice of the evaluation data analysis method can have more than a minor impact on evaluation results. If this is true in the general case as well, this has strong implications on the way evaluation data should be handled in COTS selection cases.

The case study reported in this paper provided initial results and practical feedback on main aspects of the OTSO method. It seems that the OTSO method addresses important and often ignored problems in COTS usage. However, due to the limited number of data points, i.e., evaluators and cases, the results are not conclusive. We plan to carry out additional case studies and experiments to validate our method further.

7.3 The Process to Identify and Specify Reusable Patterns

It is widely believed, in the software engineering community, that the primary benefits associated with a reuse program are higher software quality and faster cycle time. These benefits, however, come when the reuse maturity grows from simple reuse of code to reuse of high level designs and architectures. As Christopher Alexander, an architect, formulated it, the idea is to make available to the designers a set of solutions to recurring problems which have been proved useful and reliable in other cases. These solutions are designed by experts in a particular application domain and then used by non-experts.

Design patterns are mini-architectures which represent recurring solutions to a design problem within a particular domain. They facilitate architectural level reuse by supporting the definition, composition, and evaluation of key components in a software system. A software architecture describes how a specific system is decomposed into components, how these components are interconnected, and how they interact with each other by either passing information or executing services. At architectural level, a large amount of experience reuse is possible and for this reason the reuse study has been focused on this level.

In the case of the ECS system, the idea of design patterns is widely applicable to a variety of problems. Design patterns can be developed or extracted from Release A and B and applied in further releases of ECS. Higher level design patterns can also be developed and made available to the user community in order to develop specific applications interfacing with ECS. In this study the focus has been on the first application even though the experience accumulated in the study can usefully be transferred to the other application.

There are two levels of abstraction to be documented in the specification of design patterns. The first level is the architectural one, where design patterns are specified according to their function and role within a system. The specification of this level, an example of which has been presented in Section 5, is called architectural design pattern (or framework). It is expressed as a set of classes with special emphasis on functionality and interfaces. The second level specifies the internal structure of the architectural design pattern, the roles of the component classes, the mutual interfaces. This level contains the guidelines for implementing the architectural design pattern through the coordinated interaction of lower level design patterns, each of them reusable in different contexts. The reuse study described in this paper has focused on the first level and intends to work on the second in the current year.

This subsection outlines the process used to derive the architectural design patterns shown in Section 5 of this report. This process, which is repeatable and can be replayed at periodic points with updated information, is made of the following phases and is shown in Figures 7-4 and 7-5:

1. SCOPING

    - The context chosen in ECS is the Science and Data Processing Segment (SDPS) and its CSCIs. The level of detail used is the one offered by the Release A Critical Design Review Specifications and Release B Interim Design Review Specifications.

    - The first activity is aimed at defining the scope of the study by choosing one or more subsystems and subsystem components which have enough interaction among each other and well identified interfaces with the rest of the system to be seen as building blocks of the subsystem.

2. OBJECT MODEL ANALYSIS

    - Once the boundaries of the study has been identified and specified, the object model of the components of interest is analyzed in order to find abstractions which represent possible solutions to problems encountered everywhere in the project. The easiest way to identify the abstractions is by a bottom-up approach grouping together classes and groups of classes which concurrently provide some specific functionality. The result of this analysis can be expressed as a set of class categories simply associated by a very general use relationship.

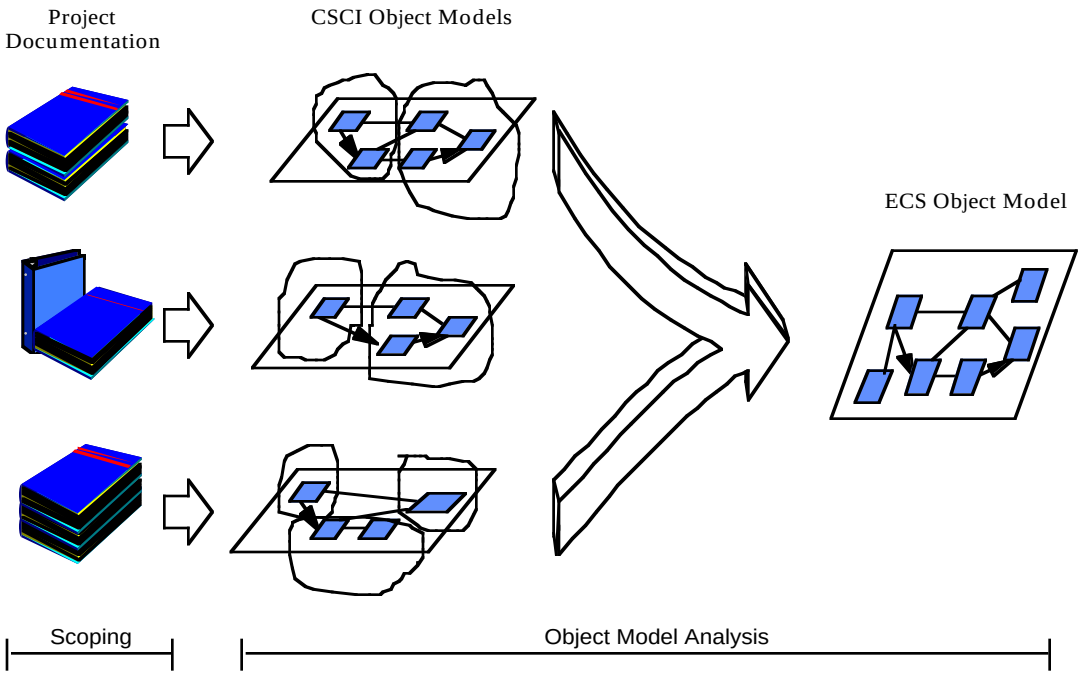


Figure 7-4. First Domain Analysis Steps

3. ADVANCED SCENARIO DEVELOPMENT AND APPLICATION

   - An advances scenario represents the specification of a direction for the evolution of the system according to well respected domain experts. It is a way to show what use is expected for the future versions of the system. The advanced scenarios are applied to the abstractions identified on the object model in the previous step to verify their correspondence to the functional needs of the system and to identify critical design patterns.
   - Use cases derived from current and advanced scenarios for the ECS program, as described in Section 4, have been applied to the abstractions identified on the object model and the following design patterns have been identified:
     - Request-Transaction-Session-Server-Result
     - Query-Search Server-Data Definition
     - Data Collection-Data Definition-Advertisement
     - Data Collection-Data Type-Search Server
     - Request-Subscription-Event Result-Notification
     - Request-Agent-Server

4. DESIGN PATTERN SPECIFICATION

The design patterns identified in the previous phase are described in terms of:

- Purpose: Role of the design pattern within a system and motivation for its use
- Interface: Interaction with the design pattern prescribed for whoever wants to reuse it in a context different from the one in which it was developed
- Structure and participants: Classes and objects participating in the design pattern and their specific responsibilities
- Origin: Where in the existing system the pattern originates

SYNOPSIS OF THE METHODOLOGY

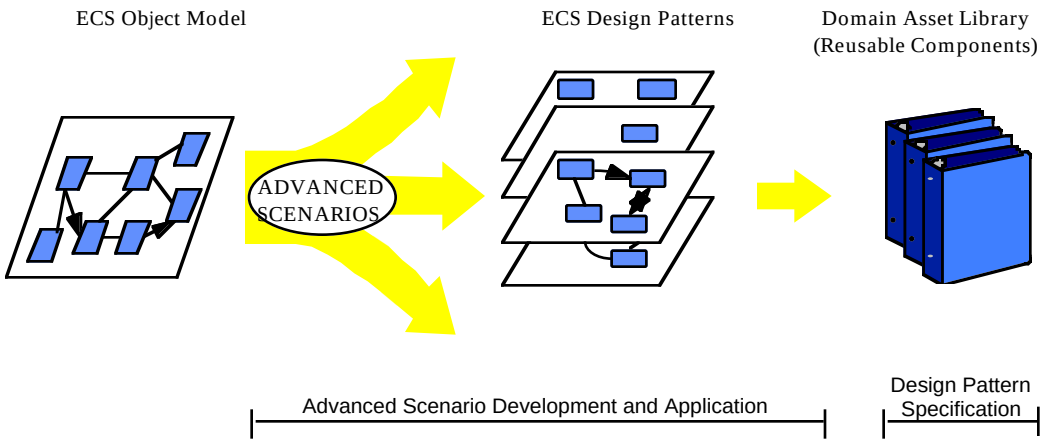


Figure 7-5. Final Domain Analysis Steps and Reuse Reengineering

An example of a design pattern is:

Name: Query-Search Server-Data Definition-Advertisement

Purpose: This pattern models the interaction of a requester object with a search engine and a supporting data dictionary

Interface: The pattern is activated by an incoming request

Structure and participants: See Figure 7-6.

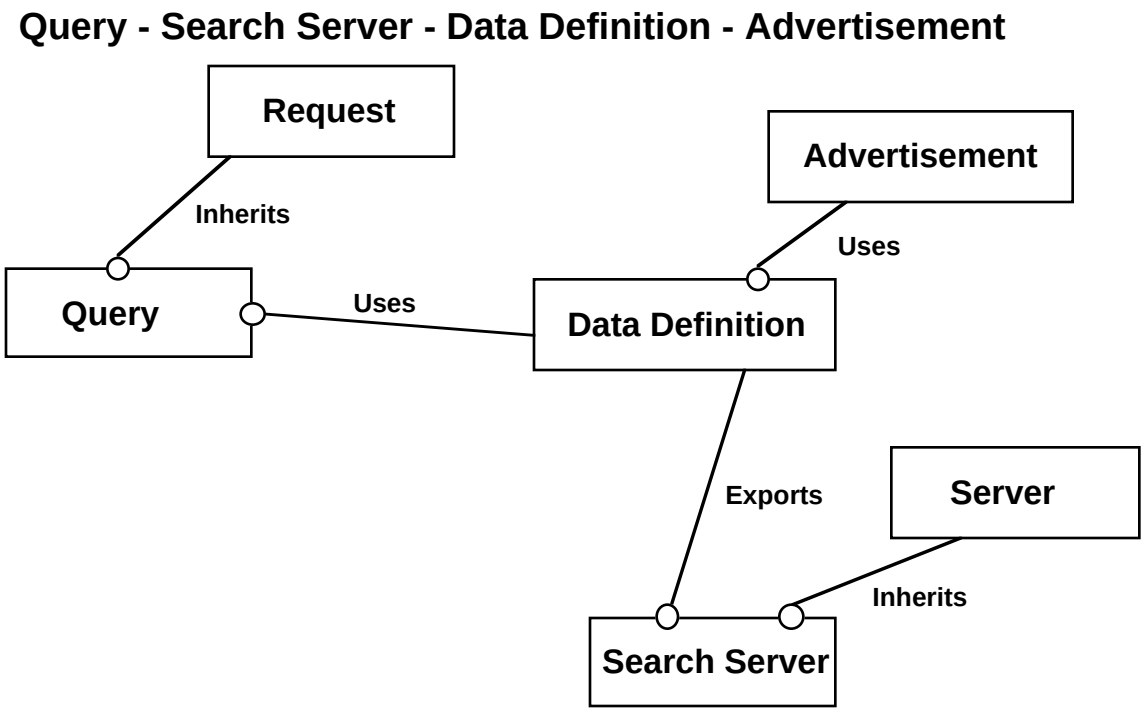


Figure 7-6. Query - Search Server - Data Definition - Advertisement

Origin: The pattern is partially implemented in the following ECS components:

Data Dictionary Service

Query Service

Search Request

This page intentionally left blank.

8. Conclusions and Recommendations

8.1 Conclusions

A potential transition from the Release A and B ECS architecture into an evolutionary change era may involve a more open and decentralized model. The advanced scenarios typified in Section 4 strongly suggest this evolution.

The contribution of the reuse study to this general picture is to show what is a reference model of the ECS architecture and how such model can be derived. The short term, immediate goal is, however, to prove feasibility and value of an approach to software reuse based on architectural concepts. The leading idea of the study is that, if we base reuse on a domain specific architecture, we achieve a better cycle time in satisfying the evolving needs of the user community, and also obtain the desired model of the core system that provides intellectual order to future concurrent engineering efforts. Cycle time reduction is achieved by selecting and using “large” reusable software components well suited to the needs of the application domain. The model of the system is obtained by looking at domain specific abstractions and searching for design solutions that seem widely applicable.

As we stated in this report, the goals of the reuse study are very ambitious and can necessarily be addressed only in sections. In 1995 the study focused on:

- Selection of reusable components: An approach has been developed for deriving and applying evaluation criteria for reusable software components. The approach is based on a taxonomy of factors that influence the selection of the components and on a hierarchical decomposition method used to transform the reuse goals into a hierarchy of evaluation criteria.
- Specification of a reference architecture: A methodology has been developed for reverse engineering a software architecture from the design documentation, and extracting important and reusable design patterns. The methodology has been applied to some ECS subsystems.

The major lessons learned in the study are:

- It is definitely possible to consolidate some of the best practices in reusable component selection into a coherent and usable methodology.
- The use of the methodology improves the efficiency and consistency of selection.
- To make the reusable component selection process repeatable, an organization needs to build up, besides software process maturity, a reuse maturity which includes, among other things,
  - specification of the selection process and of the selection criteria,
  - organization support: resource allocation, incentives, management support,
  - search support, and

- evaluation support.

- It is possible to use a conceptual software architecture to identify promising design patterns to be used in future increments of the system.
- It is possible to develop and use a methodology to derive the conceptual software architecture and associated design patterns.
- To make reuse-in-the-large, it may be necessary for an organization to build up its reuse process by optimizing it in specific domains.

In conclusion, although many areas of software reuse still need to be explored, the reuse study has yielded in 1995 some interesting products and valuable lessons learned.

8.2 Recommendations

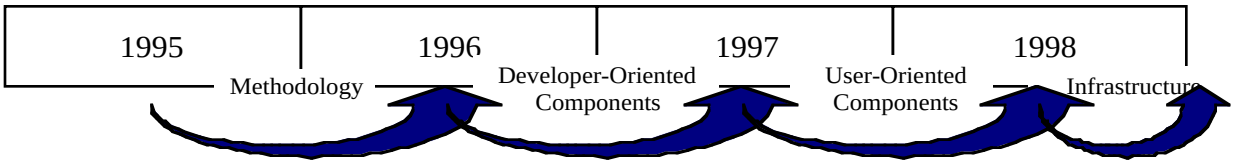
The experience gathered in 1995 through the study described in this paper should be made available to the whole ECS community in order to support the development of both a new, federated solution to the information needs of the Earth Sciences community, and an application framework relying on such federation.

Therefore the continuation of the reuse study will focus on the specification of architectural design patterns to be used by whoever wants to interact with the ECS system, or parts of it, without specifically knowing the details of the ECS implementation. As we said in Section 7, an architectural design pattern is a reusable design of a program or a part of a program expressed as a set of classes and objects. It is a mixture of concrete and abstract notions all combined into a use metaphor, i.e. a way to verbally and conceptually represent the pattern. An example of metaphor commonly used for retrieval systems is the library metaphor in which the access to information is modeled as the behavior of an individual using a library (with catalog, cards, sections, shelves, etc.).

Architectural design patterns are designed by experts in a particular domain (e.g., the designers of ECS) and then used by non-experts (e.g., an earth scientist who wants to develop a model based on ECS-extracted data). The principal audience of the continuation of the study will be not only the community of the ECS developers but also the community of those who want to solve typical problems associated with the use of ECS.

The reuse program will develop and specify in a reusable way architectural design patterns trying to assess how well they work to serve the needs of both the developers and users communities.

This part of the program is very much in tune with the overall purpose of the Experience Factory which is not only to assess the impact of a specific technology, but also to package that technology and the associated experience for future reuse. While the assessment of the impact of a reuse-in-the-large technology is still ongoing, the first steps aimed at packaging that technology and the resulting products are being taken. In the ECS case, because of the open nature of the system, this activity will try to address the interests of both the developers and the users of EOS.



1995	1996	1997	1998
Goal: • Define a methodological framework for large scale reuse	Goal: • Experiment with the methodological framework for large scale reuse	Goal: • Expand pattern based approach to user community	Goal: • Develop support infrastructure for ECS design patterns
Activities: • Assess existing technologies • Develop reuse approach	Activities: • Identify and specify reusable design patterns within ECS • Define and use reuse measures	Activities: • Analyze ECS use cases • Identify and specify user relevant reusable design patterns • Define use framewrok	Activities: • Develop use processes • Develop support processes
Products: • Process to Select Reusable Components • Process to Identify and Specify Reusable Patterns	Products: • Catalog of important design patterns with full specifications and ECS derivation • Improvements and updates to the methodology	Products: • Catalog of user design patterns • Frameworks for incorporation into end-user systems	Products: • Operational guidelines for the reuse support infrastructure

Figure 8-1. Overview of the ECS Reuse Program

Figure 8-1 summarizes the goals of the ECS Reuse Program as they have been presented in this section. The goals are categorized by year and associated with activities and products. The basic idea is to apply the methodological concepts of this report to some critical areas of the ECS system. The results of this exercise should be essentially two products:

- A catalog of design patterns internal to the ECS system and reusable across several subsystems. Each element of the catalog will be a fully specified architectural design pattern with directions to complete it and execute it.
- A similar catalog of user design patterns, i.e. architectural design patterns to be used by the implementors of applications which rest on the ECS services.

Together with these products the program will deliver operational guidelines for maintaining and evolving them and for supporting the developers in the user community

This page intentionally left blank.

Appendix A

Appendix A is an oversized drawing and is not available electronically.

This page intentionally left blank.

Appendix B

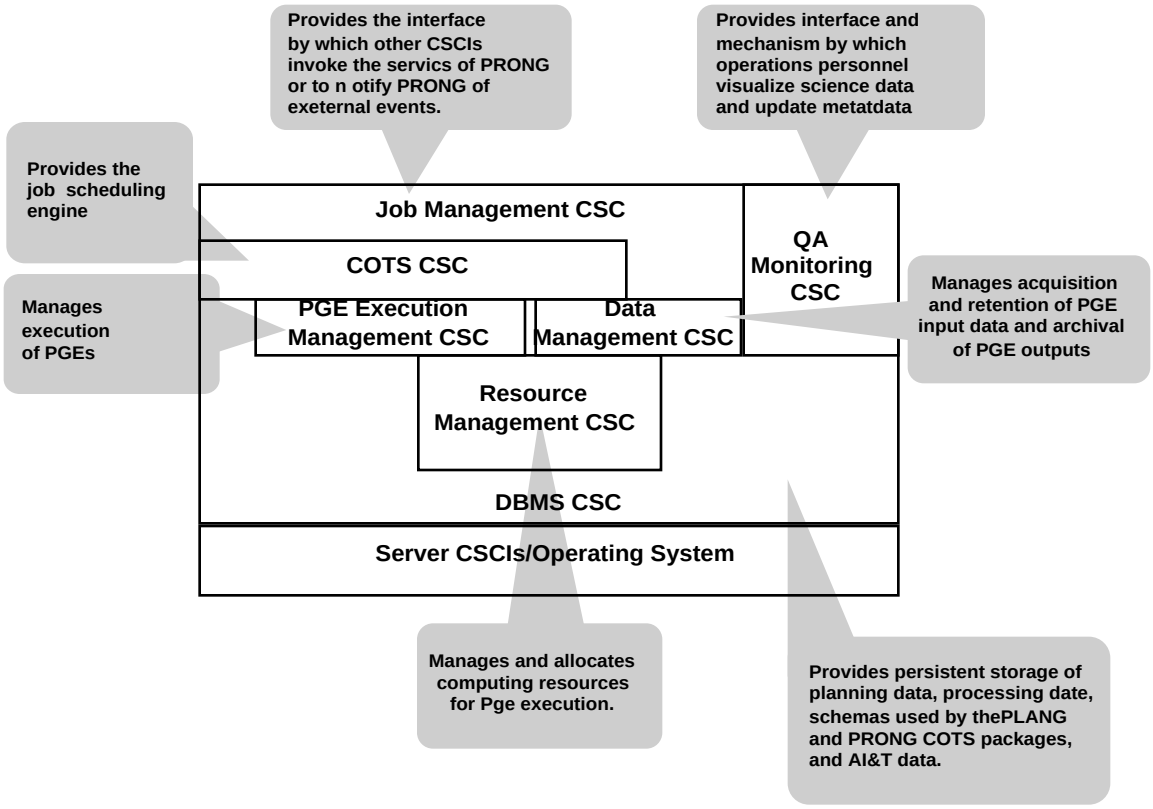


Figure B-1. Processing CSCI

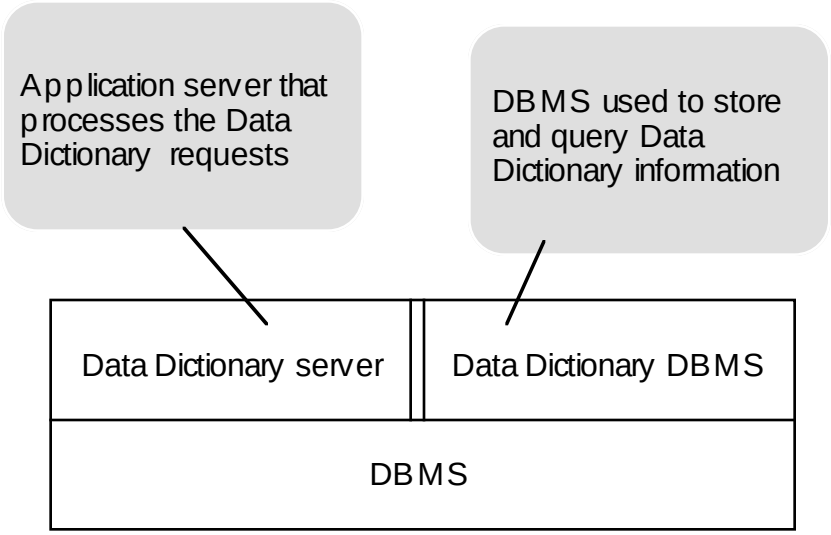


Figure B-2. Data Dictionary CSCI

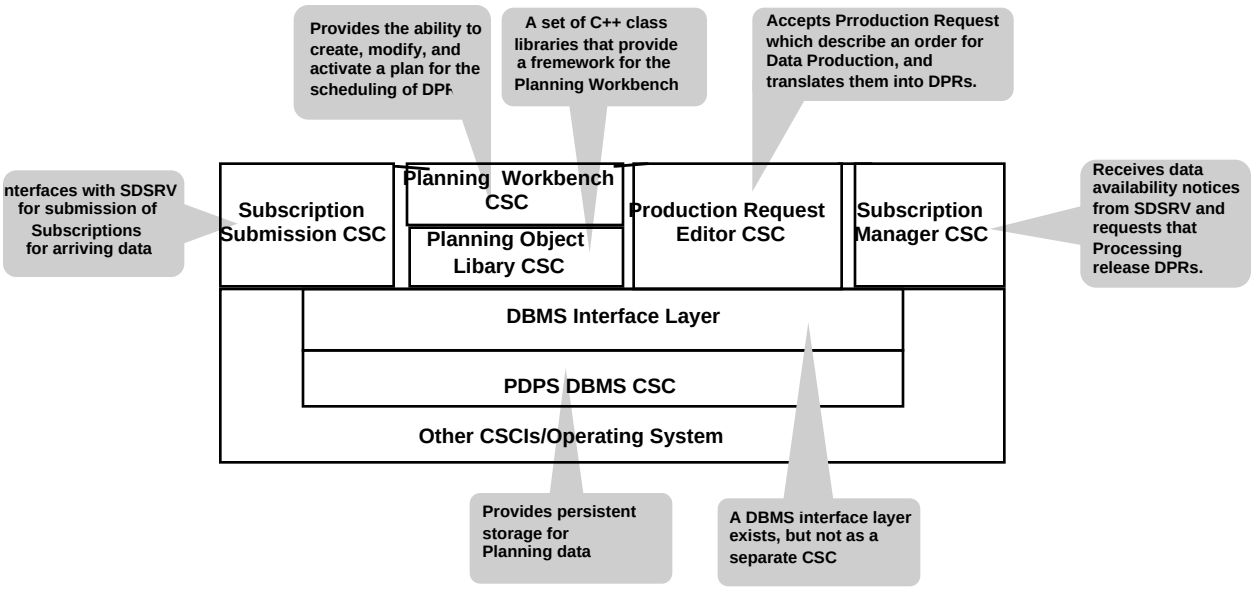


Figure B-3. Planning CSCI

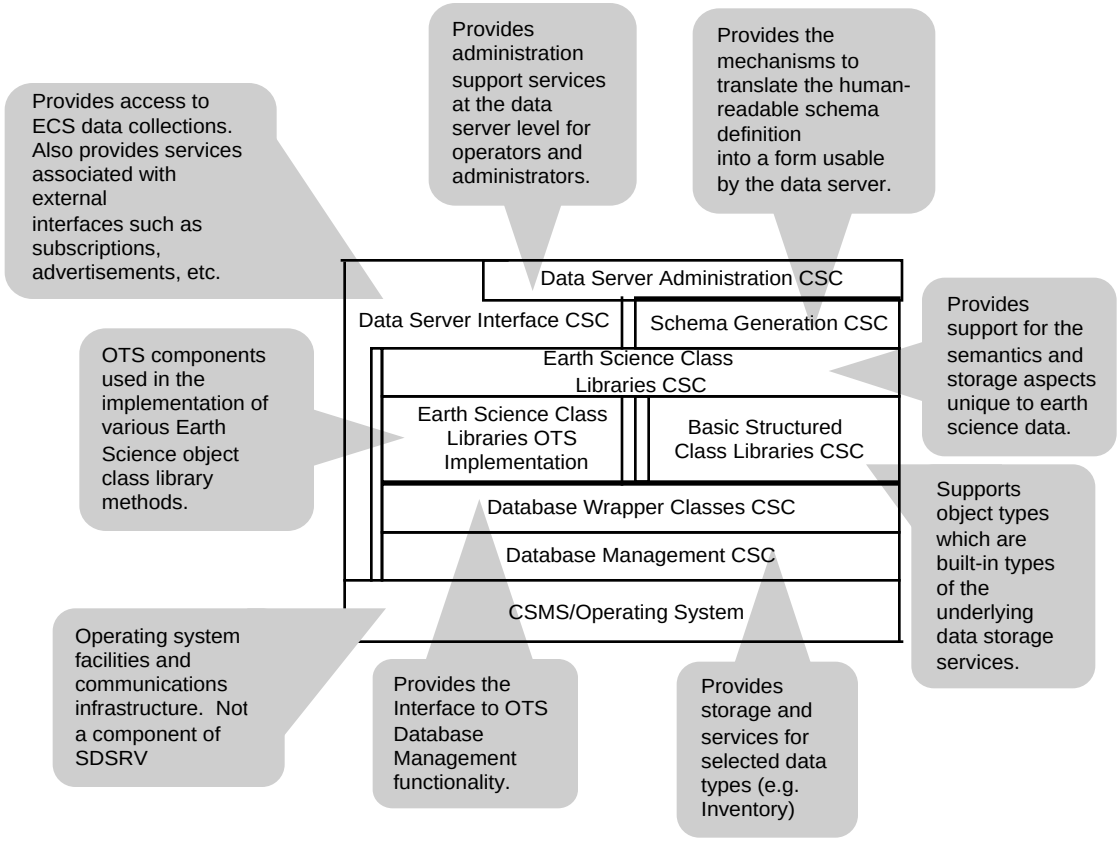


Figure B-4. Science Data Server CSCI

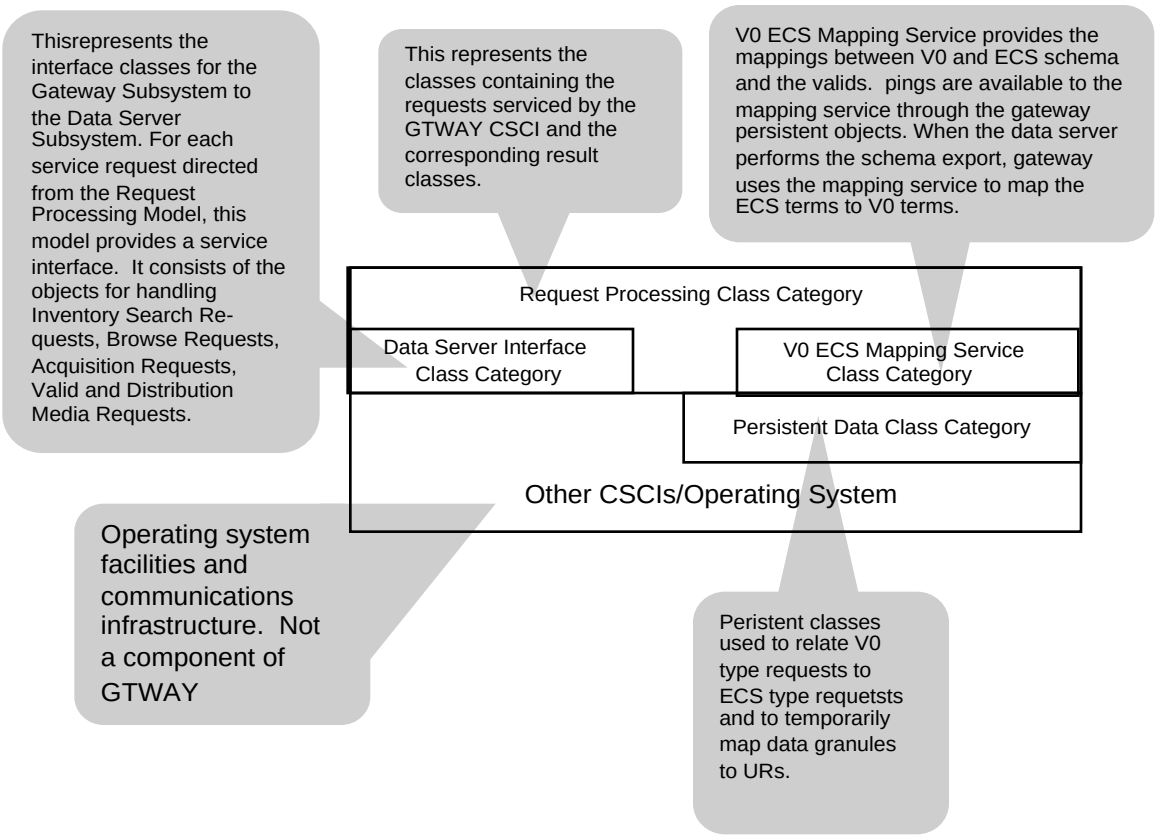


Figure B-5. V0 Gateway CSCI

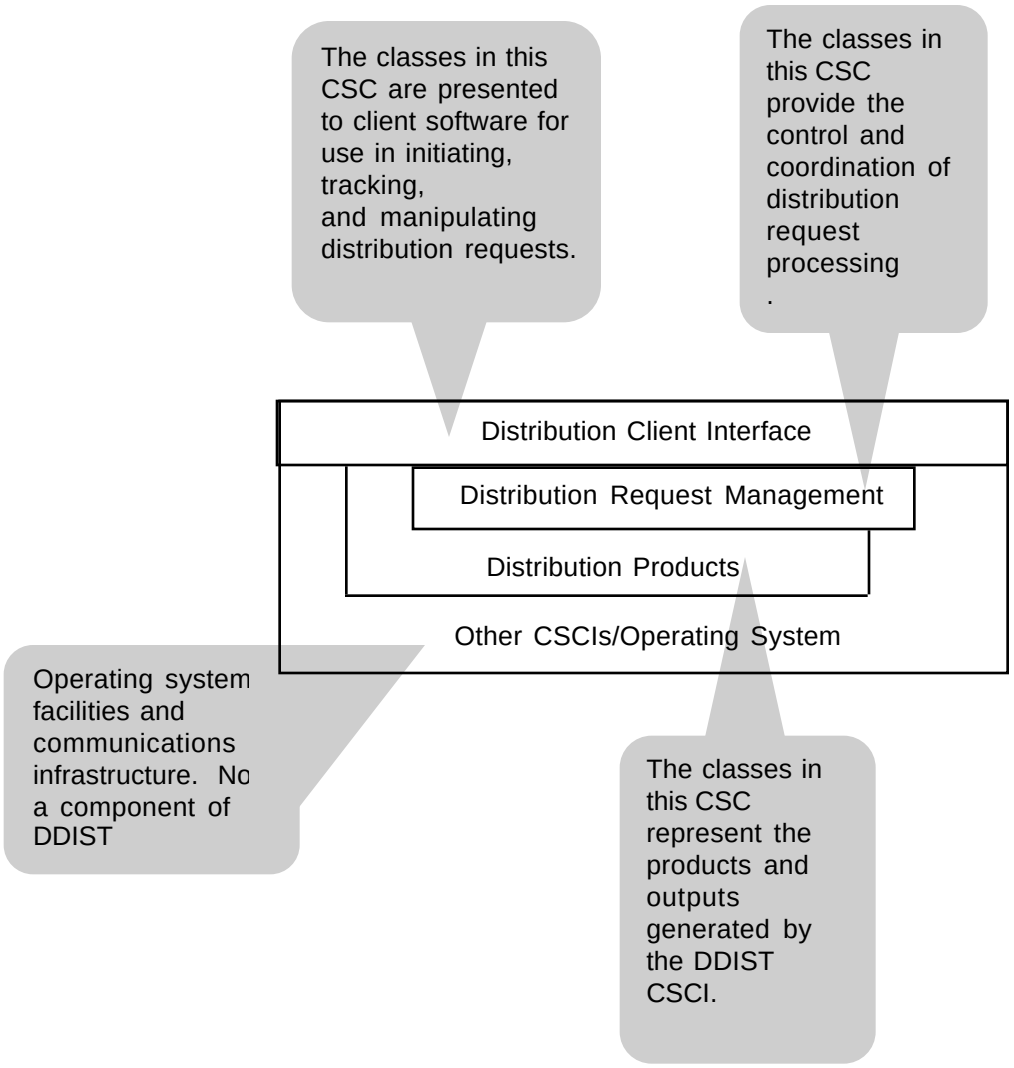


Figure B-6. Data Distribution CSCI

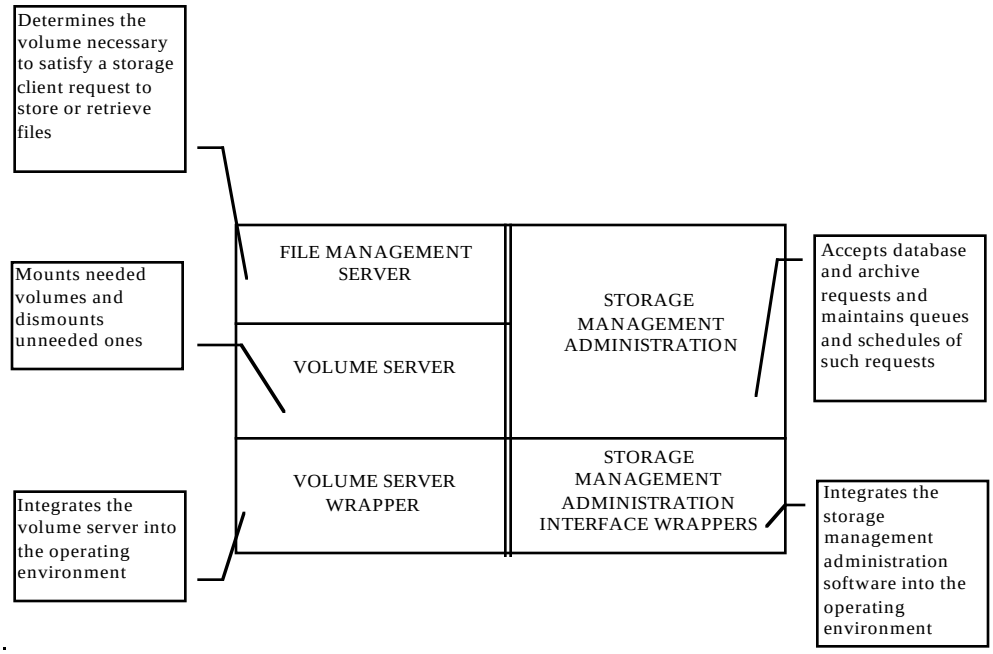


Figure B-7. Storage Management CSCI

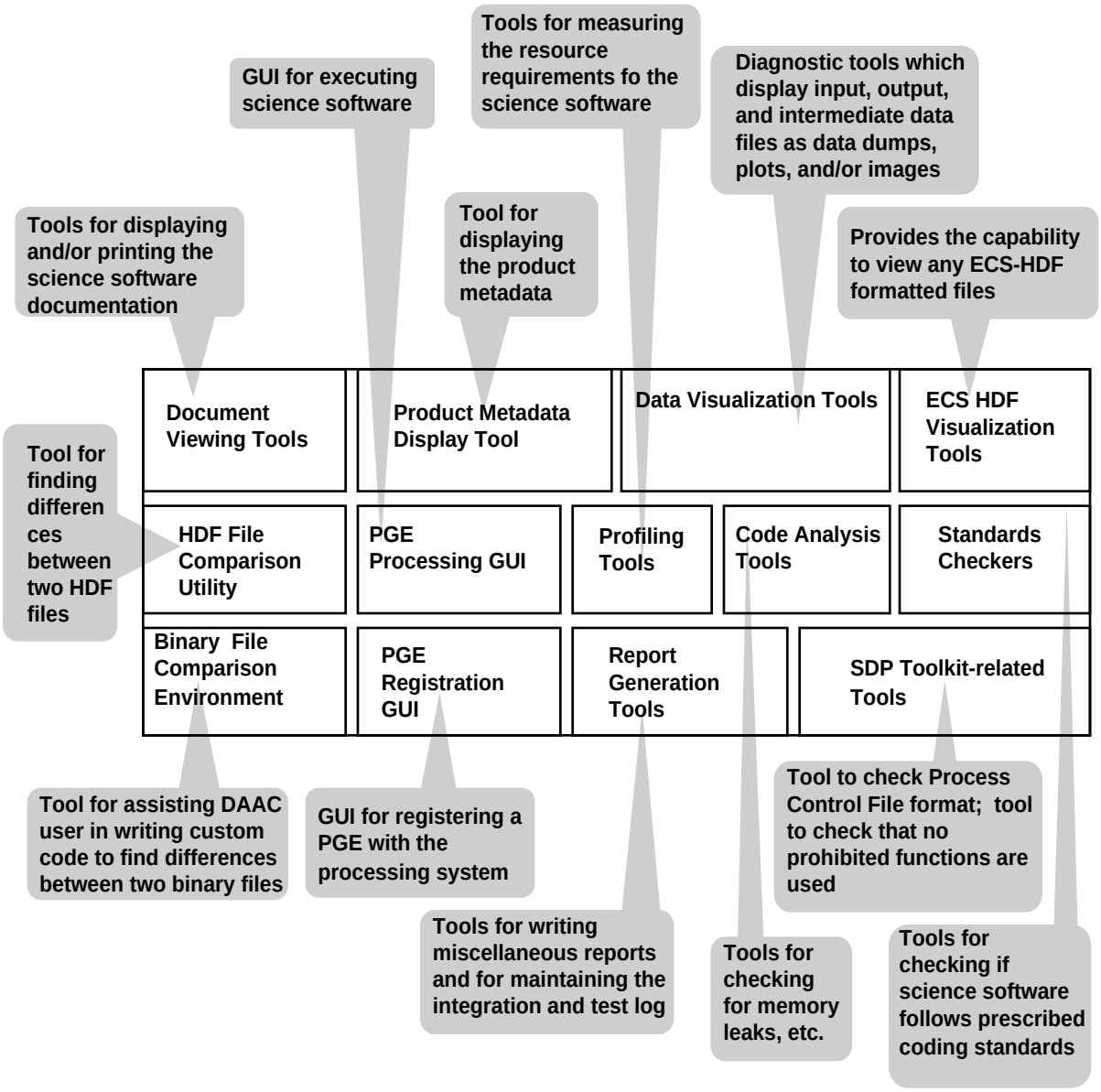


Figure B-8. Algorithm Integration and Test CSCI

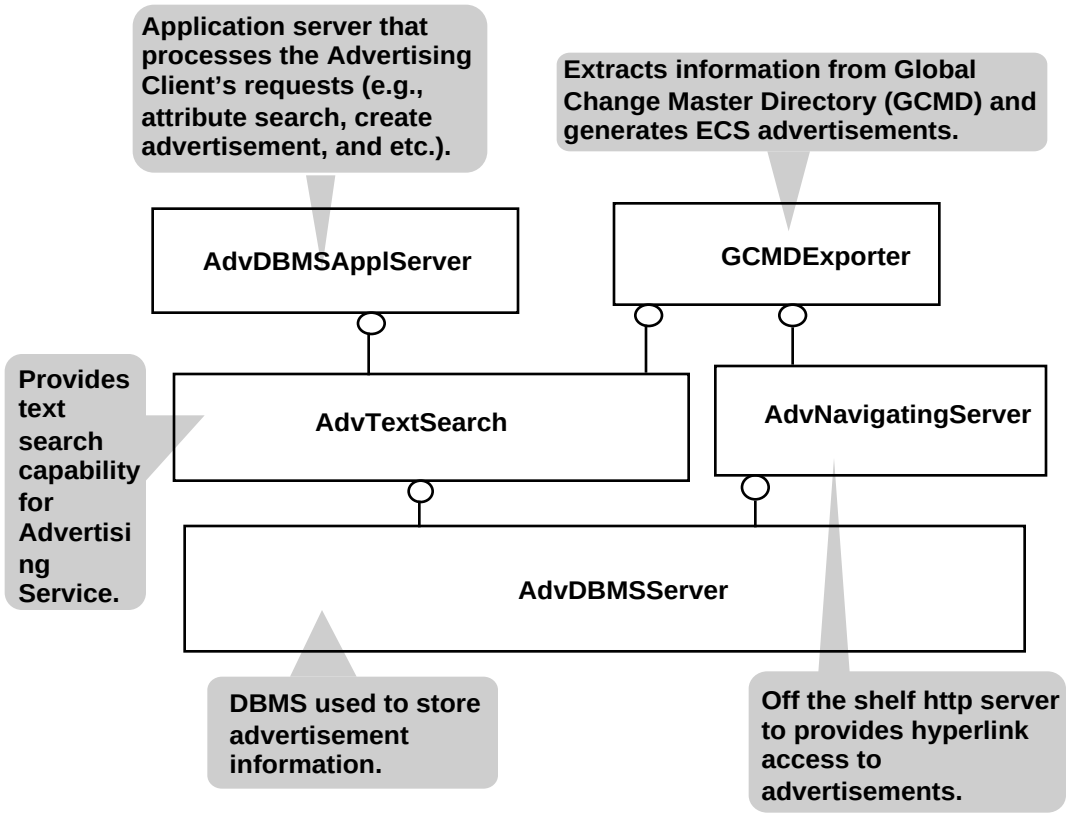


Figure B-9. Advertising CSCI

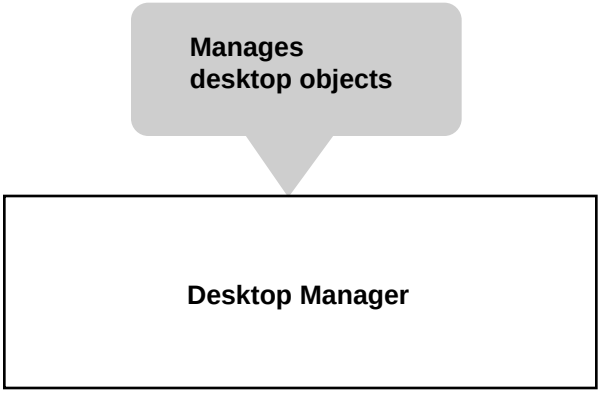


Figure B-10. Desktop CSCI

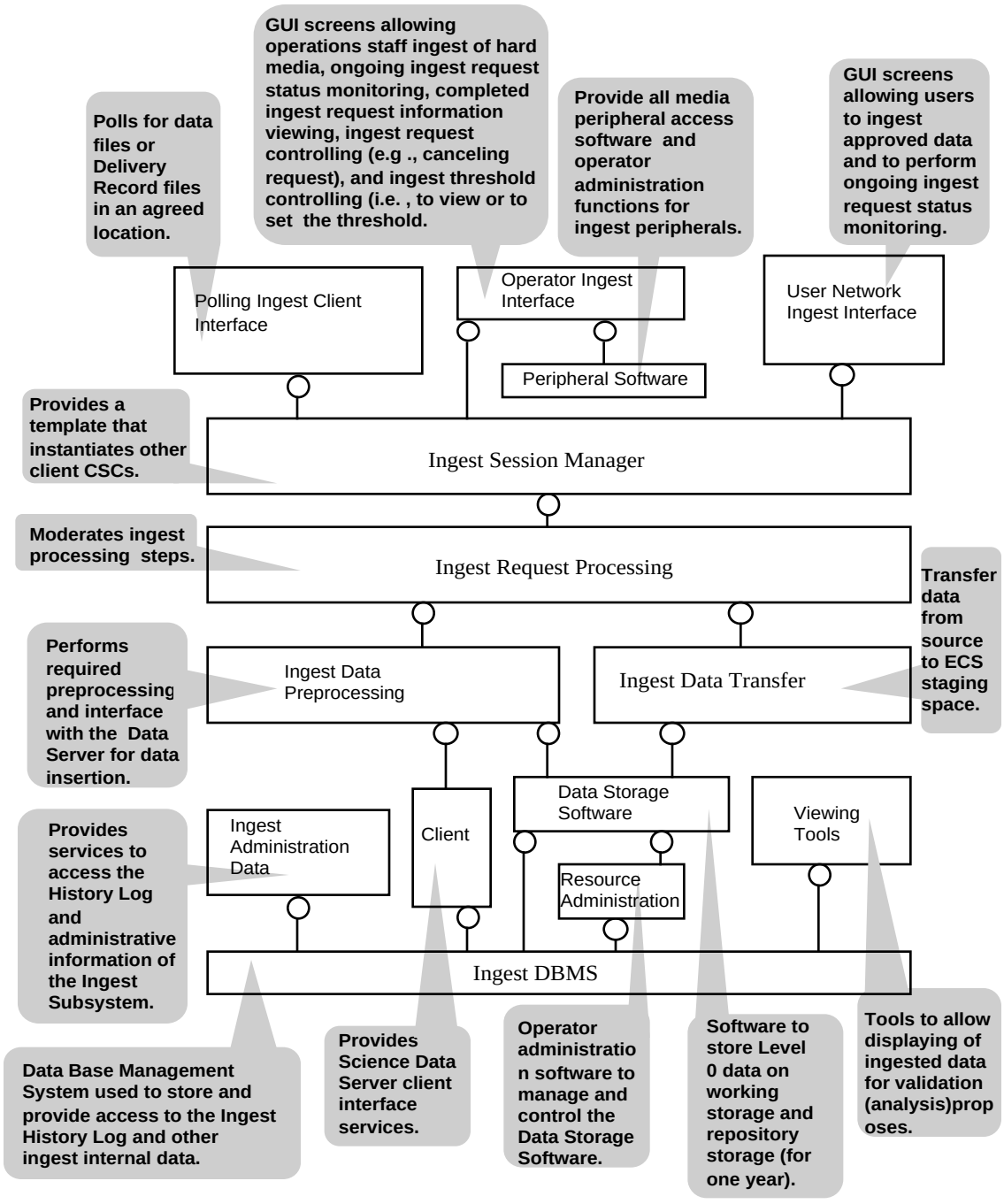


Figure B-11. Ingest CSCI

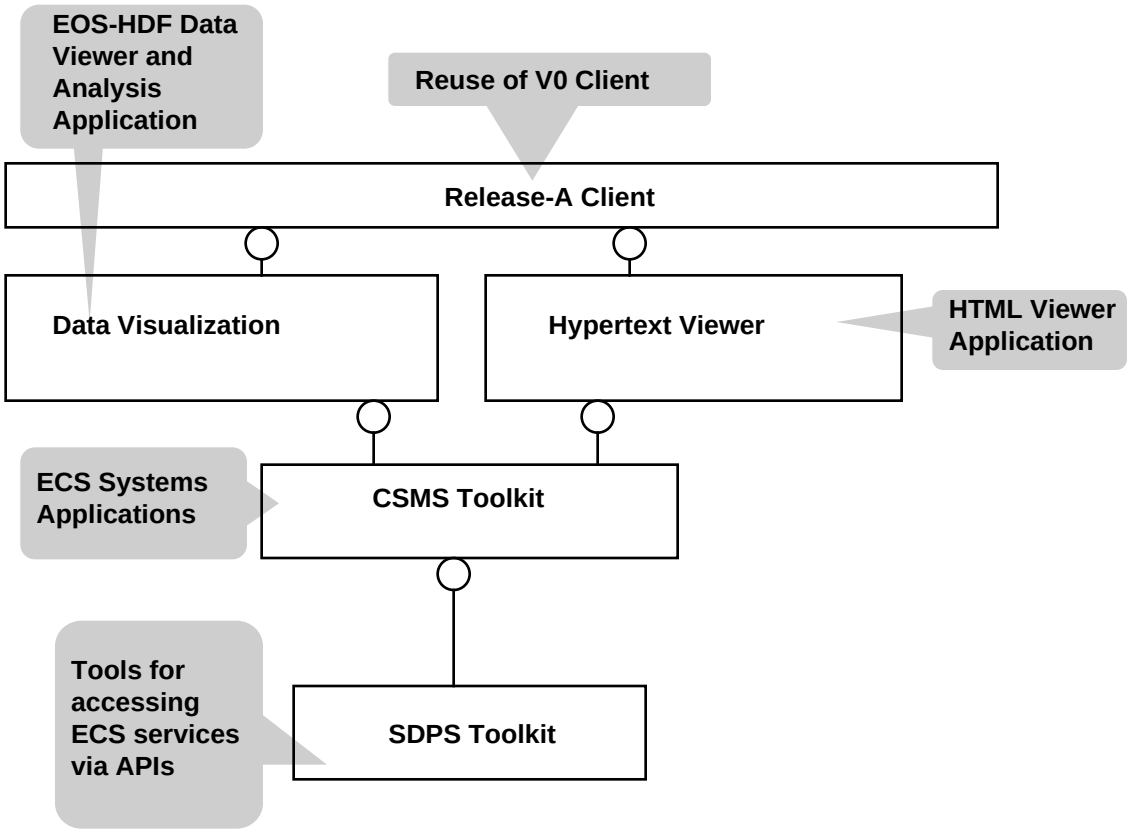


Figure B-12. Workbench CSCI

This page intentionally left blank.

Abbreviations and Acronyms

AHP	Analytic Hierarchy Process
API	Application Program Interface
BOE	Basis of Estimate
Co-I	Co-Investigator
CORBA	Common Object Request Broker Architecture
COTS	Commercial Off the Shelf Software
CSC	Computer Software Component
CSCI	Computer Software Configuration Item
CSS	Communications Subsystem
DBMS	Database Management System
DID	Data Item Description
ECS	EOSDIS Core System
EOSDIS	Earth Observing System Data Information System
FOS	Flight Operations Segment
HCL	Hughes Class Library
LAI	leaf are index
NPP	net primary productive
OTSO	Off the Shelf Option
PDA	personal digital assistants
PI	Principal Investigator
SCDO	Science and Communications Development Organization
SCF	Science Computing Facilities
TRMM	Tropical Rainfall Monitoring Method
VAP	Value-Added Providers
WSM	weighted scoring method

This page intentionally left blank.